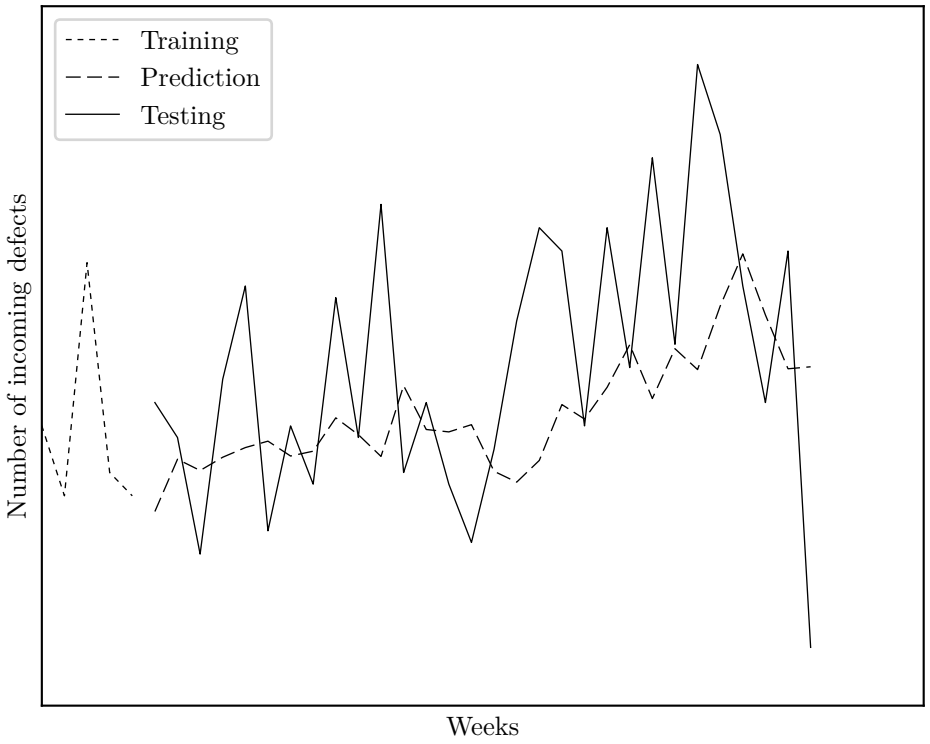

Applying machine learning to key performance indicators

Master's thesis in Software Engineering

MARCUS THORSTRÖM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Applying machine learning to key performance indicators

MARCUS THORSTRÖM

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Applying machine learning to
key performance indicators
MARCUS THORSTRÖM

Supervisor: Miroslaw Staron, Dept. of Computer Science and Engineering
Examiner: Jan-Philipp Steghöfer, Dept. of Computer Science and Engineering

Department of Computer Science and Engineering
Software Engineering Division
Chalmers University of Technology and
University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Defect inflow predictions

Gothenburg, Sweden 2017

Applying machine learning to key performance indicators
MARCUS THORSTRÖM
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Background Making predictions on Key Performance Indicators (KPI) requires statistical knowledge, and knowledge about the underlying entity. This means that a measurement designer needs to do manual work to define and deploy the KPIs. As the use of machine learning has become increasingly popular, computing power has become cheaper and more accessible, we can replace manual assessments with automated algorithms. Using the predictive power of machine learning to predict KPIs is a natural step in this direction.
Objective This thesis investigates three different KPIs in two different domains; it explores how to apply machine learning in predictions of these KPIs. The KPIs are defect inflow and defect backlog of a single product at Ericsson AB and the status level of parameters used in car projects at Volvo Car Corporation (VCC).
Method The method is divided into six research cycles where all three KPIs are investigated using different aspects and methods. The two main methods used is a linear regression approach and a rolling time frame. The linear regression method is applied two times to different aspects of the status KPI at VCC. The rolling time frame is applied to all three KPIs investigated in the remaining four research cycles.
Result The result shows a relative error of 12% when applying the linear regression approach to the status KPI from VCC and 24% when predicting each status level by itself using the linear regression approach. The rolling time frame showed that the best prediction for predicting one week ahead is to use the previous value as this gives an error of 1%, both when predicting the average status and each status level by itself.
The defect inflow predictions showed an error of 19% when applying a KNN algorithm to the rolling time frame.
The defect backlog yielded an error of below 1% when using the previous value as a prediction.
Conclusion The inflow predictions was the only predictions that proved better then previous attempts in literature, but as this was only applied on a single product in a single company this is not generalizable. It does provide a new way of predicting the defect inflow not previously seen before. The result from the linear prediction at VCC reviled a way of working which was desirable for the organization as the linear reporting was a ideal goal to strive for.

Keywords: Key performance indicators, Machine learning, Supervised learning, Defect inflow, Defect backlog, Defect predictions.

Acknowledgements

I would first like to thank my supervisor prof. Dr. Miroslaw Staron for his help and guidance when writing this report. His broad knowledge of the subject and quick interactions guided me in the completion of this report.
Second, I would like to thank both of my industrial supervisors Björn Andersson and Wilhelm Meding for providing me with data, information and a place to work. The experience working with two very different companies has been very useful and will provide good insight in my future working life.
Lastly, I would like to thank my partner My Huttunen, for the help and support during the last six months. As well as the support during my entire time at the university.

Marcus Thorström, Gothenburg, June 2017

Contents

List of Figures

List of Tables

1
Introduction

Large organizations developing software use Key Performance Indicators (KPI) to measure the ongoing process and to understand the state of the project [5]. KPIs are also used to increase the performance of the organization by measuring the success [6, 7, 8]. Organizations, such as Volvo Car Corporation (VCC) and Ericsson AB, rely on KPIs to understand, monitor and plan the process of development. In these cases, statistical analysis and experience are two main aspects when forming a KPI, which can be a problem if the experience is not sufficient.

Forming a KPI takes a lot of time and resources as the KPI has to be well constructed to not cause any unwanted effects [9, 10, 11]. Acting upon a KPI is as also a crucial task for success for large organizations.

A topic that has become increasingly interesting is Machine Learning (ML). As computational power has become cheaper and more available, ML has reached a more important role in the research field of Software Engineering. Previous research has used ML to tackle well known problems in Software Engineering, such as estimating software development effort [12] and defect prediction [13].

As the theory of KPIs has been extensively studied [14], forming a KPI based on ML is a natural choice. Since ML can help detect underlying patterns in the data, applying ML to KPI can support the decision making of stakeholders. Another reason why ML is suitable to be used in KPI research is its ability to perform predictions on data, which is useful for monitoring the current trend and observe deviations.

This thesis intend to apply ML predictions to existing KPIs in industry to make predictions on the raw data to reveal underlying trends. These predictions can then be used in the development process to govern future work.

Other researchers has begun using ML to tackle various problems in their field. A few examples are: Astronomy [15], Bioinformatics [16], Radiology [17] and Economics [18].

1.1 Problem statement

In most industries today, KPIs are used to represent the current status of a process [19]. This applies to Software Engineering as well, as a major part of development is monitoring the status and revising a strategy. The problem is to create a correct representation of the KPI on an aggregated level and not lose too much information in trends not visible at the aggregation level. If machine learning could be applied on all raw data and show hidden trends that disappear in the aggregation of the

data, these trends could be used to govern the development process to more effective software development. Using ML predictions, these trends will become visible and can be used in software development by giving new insight to the process of developing software.

To test this theory, data from VCC and Ericsson AB will be used.

1.1.1 Volvo Car Corporation

At VCC, the software department of the propulsion unit is investigated. Inspecting various KPIs used in this department, one particular is interesting. This KPI is relates to the parameter calibration status of a project.

Here, an identified problem is that a large amount of parameters used in the software needs to be calibrated to optimal performance of the constructed component. Since the software engineers do not calibrate and the calibrators do not write code, a system is used to alter parameters without having to alter source code directly to make the calibration work easier. The development process of parameters involves status labeling of each parameter, divided into 11 degrees of readiness. This readiness scale consists of numbers, to be able to create an average value easily. The KPI represent the average value each week, as this reflects the current status in a project. In the process of classifying the status, three different stakeholders are involved: the programmer, the calibrator and a project manager. The programmers task is to implement the code and add each parameter to the system. The calibrators task is to test and change the parameter to determine the optimal value. The project managers task is to take responsible for the process and determine when each parameter is production ready. There are cases when the programmer can actually calibrate the hardware without the use of a calibrator, in this case only two stakeholders are involve.

Today, an average value of the status of each parameter is displayed on a monitor, together with pie-charts of the distribution of the degrees. These graphs indicate the status of the project and governs how to spend the resources. Since this level is aggregated to the week and average status level, a lot of underlying information is averaged out.

A desired outcome from applying machine learning would involve a prediction on how the average status will look in the future, together with pie-charts on the predicted distribution. As these predictions can aid the decision maker in the process of planning. A special desire in the predictions is to find underlying patterns which makes predictions accurate and reliable.

1.1.2 Ericsson AB

Ericsson AB has established a way of incorporating their KPIs in everyday use in the organization. As there are many KPIs to investigate, a stakeholder suggested the Defect Inflow rate. This is then extended by also investigating the defect backlog KPI.

Large software organizations manage their daily work by the status of the software and how the progress is evolving, one aspect of this progress is the numbers of

defects in the software. As software can never be fully fault free or fully complete, an important trade of is the amount of defects present in the software at a given time as well as the inflow rate of defects.

This information helps software organizations such as Ericsson AB to plan and schedule the allocation of developers producing software and developers fixing defects. Another viable use of this information is the estimation of when the software can be released to the market.

The defect database contains defects that have been discovered internally at Ericsson AB during development and testing.

1.2 Research Questions

This thesis intends to explore the possibilities to extend the current KPI status with prediction to support stakeholders in the decision making process. Therefore, the research questions intended to be addressed are:

Q1) What data used to form KPIs is available and can be used as features and labels in a machine learning prediction?

The answer to this research question will provide the knowledge on how to characterize the data at VCC and Ericsson AB when used for machine learning. Having the available data, the following will be investigated:

Q2) How to use machine learning algorithms to improve the current state of a KPI by predictions?

An evaluation of the predictions will also be made to estimate their relevance. Addressing this question will provide a set of potential algorithms that can be used to answer the question:

Q3) Given the available data sets at Volvo Car Corporation and Ericsson AB – which KPIs can be improved?

The result will show a method for applying machine learning to current KPIs to improve their relevance in the organization by giving estimates to rely on and plan by.

To answer the research questions, action research [20] is used, where the problem owner is VCC and Ericsson AB. To evaluate the outcome, stakeholders related to the KPIs is going to be elicited and also compared to the literature when the process of prediction is complete. This is to analyze the influence of the outcome. The iterative process will be executed as follows: First, the current KPIs will be evaluated at the company of interest, then the data for the underlying KPI will be studied. This data will be subject to the process of using ML to predict the future progress and status of the KPI. The process is illustrated in Figure 1.1. When this prediction has been completed, the prediction will be shown to the related stakeholder to evaluate its usefulness and if the prediction can support the stakeholder in the decision making process. The process of acquiring a sufficient and correct ML algorithm will also be an iterative process.

To minimize external validity threats this will be completed at two large organizations in different domains, VCC and Ericsson AB.

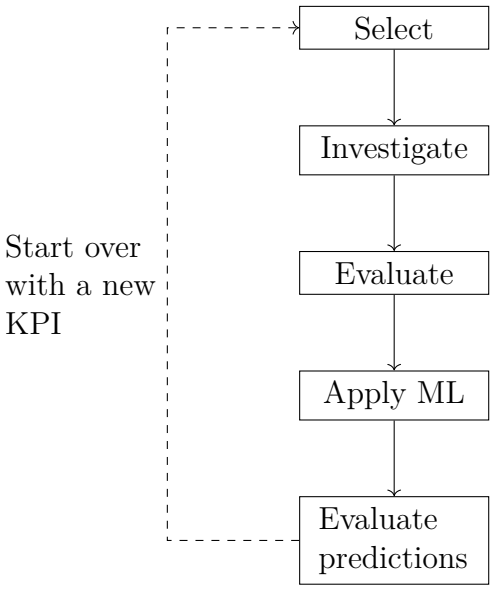

Figure 1.1: Workflow of applying ML to KPI in this thesis

1.3 Delimitation

This report will use existing and implemented machine learning algorithms and not implement or define any new machine learning algorithms. This report will use existing KPIs established at the companies and not define any new KPIs.

1.4 Report structure

Chapter 2 explains the theory behind the thesis. Chapter 3 elaborates the background of the KPIs. In Chapter 4 the execution of the thesis work will be explained, the results is presented in Chapter 5. In Chapter 6, a discussion about the result is given and Chapter 7 answers the research questions.

2

Theory

This chapter explains the notion of Machine Learning and the theory behind some common algorithms, and what KPIs are and how they are used.

2.1 Machine Learning

Machine learning (ML) is programming a computer to be able to do tasks without explicit instructions, similar to the learning process of a human or animal, by examples [21].

In machine learning the two terms *Features* and *Labels* are frequently used. In this report, a feature will represent a feature vector, which contains the known attributes of an instance. For example, using ML to estimate effort to develop software, the feature vector can be the size and complexity of the project. A label is the desired output for a feature vector in a ML algorithm. Using the same example, the actual effort of the project is the label. A simplified way of describing this relationship is $f(x) = y$ where x is a vector containing features, y is a label and f is a machine learning algorithm.

Supervised and unsupervised learning is two concepts of learning defined in ML [22]. Supervised learning in machine learning is done using feature vectors and to map these to labels with as good approximation as possible [23]. This is then applied on unseen data to map this to new labels. Supervised learning has two sub-categories, classification and regression which are explained further in section 2.1.1 respectively 2.1.2.

To use a supervised learning algorithm, a training set is required to train the algorithm to fit the specified data. The accuracy of a supervised algorithm is evaluated using a testing set. The testing set is part of the complete data set, where data points are predicted and compared to their true values to give an indication on how well the algorithm performs [3].

Unsupervised learning is used to discover patterns, trends and similarities in the multidimensional data sets effectively. The distinction from supervised learning is the lack of labels. One example of unsupervised learning is clustering [23]. Clustering groups data with similar characteristics into groups to reveal similarities and relationships.

2.1.1 Classification

Classification uses machine learning to predict a categorical value [24]. The categorical value must be a predefined value that has previously been seen in the data set. As classification maps to discrete classes it often has a way of handling unknown values and sometimes provides a probability estimate of selecting the correct class [23].

2.1.1.1 Example

A classification example from Software engineering is a *Flexible LOC Counter* [1]. Here, a classifier can be trained to recognize what is distinguished as a line of code, and what is not a line of code. This can then be summed up to a number of lines. An example on how this is applied can be seen in Figure 2.1. In Figure 2.1, the algorithm successfully identifies all lines of code, and identifies the empty rows as empty rows.

Figure 2.1: Screenshot from CCFlex tool from [1]

2.1.2 Regression

The outcome of a regression algorithm is a continuous value and not a discrete value as in classification [23]. In opposition to classification, regression does not have a defined range of output values and is therefore more uncertain it its outcome. A regression prediction is, depending on the algorithm, a combination of previously seen values with similar features or a function of its features.

2.1.2.1 Example

One example using regression is *Effort estimation*, by using a regression tree effort estimation is competitive to COCOMO and other effort estimation methods [12]. As the regression tree trains on a data set and establishes rules to give an estimation

of the required effort. In this example, the output is the effort required for a project depending on the size and complexity, which yields a number of man hours.

2.2 Machine Learning Algorithms

In this section, the algorithms used in this thesis is identified and their usage is elaborated. As the thesis only uses regression, classification algorithms is explained briefly.

2.2.1 Support Vector Machine

Support vector machine (SVM) is the collection of Support Vector Classifier (SVR) and Support Vector Regressor (SVR), which are used for classification and regression respectively [25]. The purpose of a SVM is to draw a *decision boundary* of N dimension through a set of feature vectors of N dimension. A decision boundary is a line separating the data set into different classes during classification. The process of finding this boundary is done by a Maximum Margin training algorithm [26], which draws a line through the training set and positions it as far away from both classes as possible.

In an SVR, an ϵ value is taken into account when fitting the data. The epsilon value introduces a tolerance to the fitting of the data, by giving the estimated model some space to variate [27]. The objective is to find a line as flat as possible and has taken the epsilon distance in account. An example of the epsilon tube can be seen in figure 2.2, every point outside this tube is penalized and adjusts the formation of the line drawn and points within does not affect the model. The penalization is done by the C-parameter which adjusts the penalty [28]. The penalization changes the model to adjust to these values. The algorithm terminates when a line with no penalization is drawn, meaning that no value is farther away from the line then the epsilon value [27].

When there are outliers in a data set and the outliers would distort the decision boundary to give a more incorrect result as the epsilon value is preferably set to a low value, the concept of *soft boundary* can be applied. A soft boundary is the user telling the algorithm how many outliers can be accepted [29, 30]. This is added in addition to the epsilon value.

The algorithm uses a *kernel* function to increase the dimension of the data [29]. The kernel function is the unique feature with the SVM, as this is used to raise the dimensionality of the data set by applying a function to the data set. This can show trends in a different dimensionality.

2.2.1.1 Kernel

The kernel is used to decide the nature of the decision boundary by increasing the dimensionality of the data by applying a function on each feature vector in the distribution to get a new distribution. An example of this is using the polynomial kernel and one feature, where the feature x is squared to represent a completely new

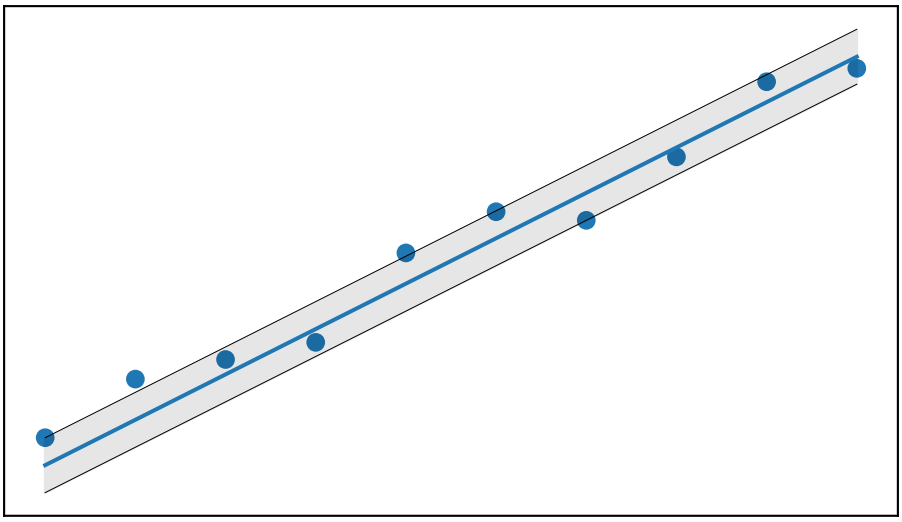

Figure 2.2: The epsilon tube of an SVR

distribution x^2. In a similar way, kernels manipulate the data to make it linearly separable. Below is a list of kernels and their parameters:

- Linear - This was the first kernel to be used in an SVM and has no impact on the dimensionality of the data as opposed to the other kernels. The linear kernel is related to a linear equation. The definition of the linear kernel is: $K(x_i, x_j) = x_i^T x_j$ [28].
- Poly - This kernel raises the dimensionality of the data by a degree. The definition of the polynomial kernel is $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$ where γ, r and d are hyper parameters set by the user [28].
- Radial basis function (RBF) - This kernel does similar work as the polynomial but to a steeper angle. The definition of the RBF kernel is: $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2)$ where γ is a hyper parameter set by the user [28].
- Sigmoid - The sigmoid kernel has a similar behavior to the RBF kernel with some parameters. The definition of the sigmoid kernel is: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$, where r is a hyper parameter set by the user [28].

2.2.2 Linear regression

In linear regression, a linear equation is fitted to a data set and minimizes the squared error between estimates and actual values. In this implementation, the ordinary least squares (OLS) solution is used where a matrix X is computed to give the best estimate. The form of the regression can be seen in Equation 2.1, where the shape of b is a $1 \times n$ matrix, the shape of a is a $n \times p$ matrix and the X is a $1 \times n$ matrix [31].

$$b = aX \tag{2.1}$$

The objective of OLS is to minimize the euclidean distance of the estimation of X as can be seen in Equation 2.2 where $||...||^2$ is the euclidean distance [31].

$$min(||b - aX||^2) \tag{2.2}$$

An example of a basic linear regression can be seen in Equation 2.3, where x is the independent variable (input), y is the dependent variable (output), β is a parameter and ϵ is the relative error [32]. Reconecting to the definition in Equation 2.1, X is the vector containing the β-parameters.

$$y = \beta_0 + \beta_1 x_0 + \beta_2 x_1 + ... + \epsilon \tag{2.3}$$

The β-parameters is used to get the best fit as possible. By changing these parameters the line will minimize the SSE (Sum of Squared Error) (Equation 2.4).

$$\sum_{i=1}^{N} (y_i - \beta * x_i)^2 \tag{2.4}$$

2.2.3 Lasso Regression

Lasso (least absolute shrinkage and selection operator) regression is an extension to Linear regression, where a shrinking parameter α is introduced [33]. This parameter is used to shrink the different β- values. The objective for the lasso algorithm is to minimize the SSE and penalty function, simplified in equation 2.5. This can result in some β-parameters being set to zero and resulting in not contributing anything to the model. This is why Lasso is useful for feature selection, as this can show that some features do not have any impact at all.

$$minimize(SSE + \alpha * \sum_{i=1}^{N} |\beta_i|) \tag{2.5}$$

2.2.4 Ridge regression

Ridge regression is similar to Lasso regression with the distinction of the penalty parameter [33, 34]. The equation for the objective can be seen in Equation 2.6.

$$minimize(SSE + \alpha * \sum_{i=1}^{N} \beta_i^2) \tag{2.6}$$

2.2.5 Elastic Net

Elastic Net is a hybrid between Lasso regression and Ridge regression. Where the first implementation, Naïve Elastic Net, is shown in Equation 2.7. Here it is clear that this is a combination of the two algorithms. As seen from Equation 2.7, Elastic net is Ridge regression when $\alpha_1 = 0$ and Lasso regression when $\alpha_2 = 0$.

$$minimize(SSE + \alpha_1 * \sum_{i=1}^{N} |\beta_i| + \alpha_2 * \sum_{i=1}^{N} \beta_i^2) \tag{2.7}$$

This was then refined due to increased bias and poor predictions [35]. The correction was to multiply the coefficients, found after minimization, by $(1 + \alpha_2)$.

2.2.6 K Nearest Neighbor

The *K Nearest Neighbors* (KNN), is an extension of a previous algorithm called *Nearest Neighbors* (NN) [36]. NN is used to classify data points by using a distance function in an n-dimensional space and selecting the closest point in distance and use this to classify the unseen point [37]. This was then extended to become a voting algorithm of the k-nearest points, where k points were listed and these voted on what class the new points belong to. This process was then extended to cover regression by using the average value of the k nearest points [18].

KNN is an example of instance based learning [38], where no generalization is made and the model only uses previously seen data points to predict new data points. This means no computation is done until a prediction is made.

2.3 Selecting an algorithm

Selecting the correct Machine learning algorithm is difficult, Scikit-learn [39] provides a visual guide for making this process easier, see Figure 2.3. This guide covers some of the basic algorithms and when to apply them. The original graph is completed with links to each algorithm in the scikit learn documentation, as this is just a remake to fit this thesis.

When starting out, this guide is a useful aid towards starting to select a few algorithms and then expanding the scope to include more and similar algorithms.

2.4 Data processing

When applying machine learning, the input data must be formatted in to feature vectors, as most algorithm cannot process values other than integers [40].

2.4.1 Time dependent

Data represented in time is dependent, this must be encoded in a respective way. Sometimes the interaction over an hour or day can be more of interest then the continuous interactions. Using one hot encoding (Section 2.4.2) can be applied to divide the data points over the course of an hour or day. If the data changes over longer periods of time, these interactions needs to be modeled accordingly. Converting the time stamps to a relative number is also an approach to use. The granularity of the number conversion depends on the time interval of the data set. If different entities stretching over a time span has different starting and ending dates it might be interesting to normalize the time vectors to start at zero.

Another way of modeling time series is using a *rolling window* [41]. The rolling window is modeled by taking n previous observations, referred to as the lag, of Y as features to be used in new predictions. A formal definition can be found in Equation 2.8.

$$X_t = [X_{t-1}, X_{t-2}, ..., X_{t-n}] \tag{2.8}$$

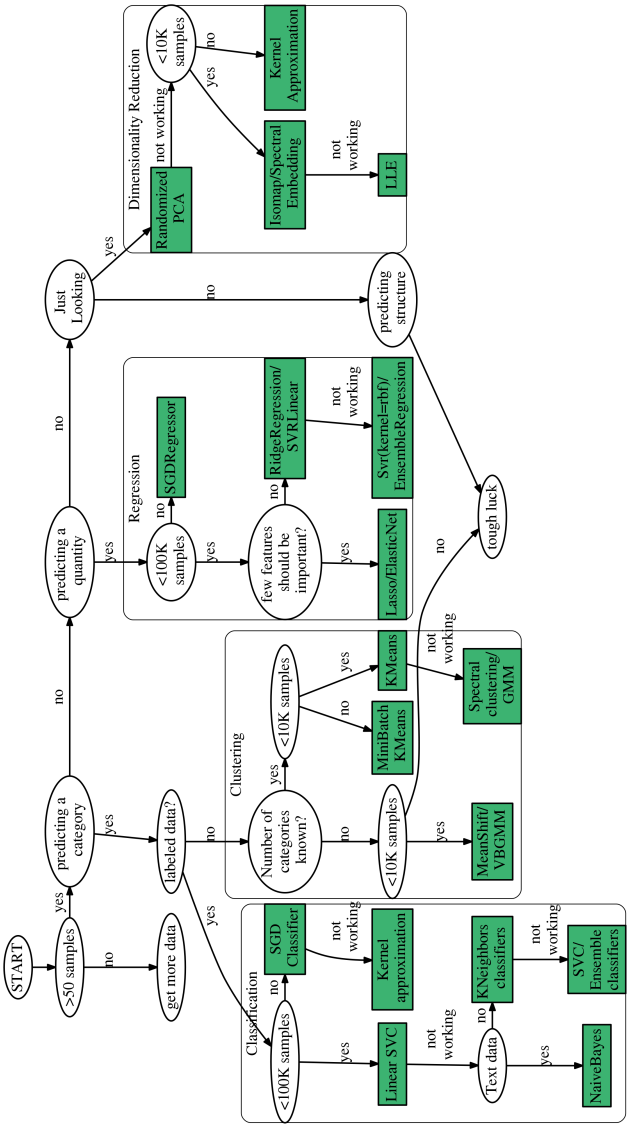

Figure 2.3: Scikit-learns guide of choosing an estimator [2].

One issue regarding the rolling window is the data is shortened by n points, as the first row must start with the n-th previous value. This can be demonstrated by an example:

A time series ordered by oldest value first is displayed in Table 2.1, where it is converted into a rolling window. It is clear that the length of the data is reduced by n points. A problem with using rolling window is selecting the n value to give a correct representation.

One downside with using rolling window is that the prediction is only able to make one prediction at a time, as the data required for x_t is $x_{t-1}, ..., x_{t-n}$. Therefore, when validating, instead of predicting 7 points ahead seven one-point predictions are made. Since the data needed for predicting the next time step is the current time step.

One aspect of working with the rolling window is that it resembles an autoregressive model in the modeling of features and can therefore be used with cross

Value
5
10
12
17
20
21

$\Longrightarrow$

t-2	t-1	Value
5	10	12
10	12	17
12	17	20
17	20	21

Table 2.1: Example of applying rolling window

validation (see Section 2.5.4) [42]. Similar to other usage of cross validation (see Section 2.5.4), it has to be applied to a training set and validated against a testing set. The selection of these sets differs from working with independent and identically distributed (i.i.d) data, as the testing set has to be appearing after the training set in a timeline, this is to resemble the real world application as much as possible.

2.4.2 One hot encoding

One hot encoding [40] is a method to model categorical features in a feature vector. The encoding is done by taking each categorical value, creating a new column and insert one in the column corresponding to the feature and zero in the rest of the columns not related to the vector. An example can be found in Table 2.2, where the categorical data has been encoded using one hot encoding.

id	category	value
1	A	2
2	B	10
3	C	3

$\Longrightarrow$

id	value	category-A	category-B	category-C
1	2	1	0	0
2	10	0	1	0
3	3	0	0	1

Table 2.2: Example data encoded using One hot encoding

2.5 Quality Criteria

To evaluate the performance of an supervised machine learning algorithm there are a number of methods.

2.5.1 Classification

In classification there is a very clear definition of a correct and incorrect prediction, therefore the scoring is intuitive. Depending on the usage of the algorithm, different scoring functions can be used. In classification, it can sometimes be more important to have false negatives than false positives, as a spam filter that lets through spam and not marks important emails as spam.

2.5.2 Regression

In regression, the predicted value is an estimate of the true value. Here an approximation is required. The measure of the approximation can take different forms.

2.5.2.1 Mean squared error

The mean squared error (MSE) is defined byEquation 2.9 [43], where y is the actual value and $\hat{y}$ is the predicted value.

$$MSE = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} (y_i - \hat{y}_i)^2 \tag{2.9}$$

The MSE is always a positive number where the best score is 0, as it indicates there is no difference between the predicted and the true value.

2.5.2.2 Mean absolute error

The mean absolute error (MAE) is defined byEquation 2.10 [43], where y is the actual value and $\hat{y}$ is the predicted value.

$$MAE = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} |y_i - \hat{y}_i| \tag{2.10}$$

MAE is always a positive number, where the best score is 0. A score of 0 indicates that there is no difference between the predicted value and the actual value. MAE, compared to MSE, gives a lower number since there is no squaring of the error. MSE, compared to MAE, cannot account for high variance in the error factor [4], as illustrated in Table 2.3, where e_i is the error, defined as $y_i - \hat{y}_i$.

Variable	Case 1	Case 2	Case 3	Case 4	Case 5
e_1	2	1	1	0	0
e_2	2	1	1	0	0
e_3	2	3	1	1	0
e_4	2	3	5	7	8
MAE	2	2	2	2	2
MSE	4	5	7	12,5	16

Table 2.3: Example illustrating difference between MSE and MAE [4]

2.5.2.3 Median absolute error

The median absolute error (MedAE) is defined byEquation 2.11 [43], where y_i is the i-th actual value and $\hat{y}_i$ is the i-th predicted value.

$$MedAE = median(|y_1 - \hat{y}_1|, ..., |y_n - \hat{y}_n|) \tag{2.11}$$

The best value of MedAE is a value of 0, meaning there is no difference between predicted and the actual value. This score can only be positive. The MedAE is not

to be confused with Median absolute deviation, which differs implementation wise [44]. MedAE cannot be used with multiple dimensions.

2.5.2.4 R^2-score

The most commonly used metrics in statistics is the R^2 score. This is implemented using Equation 2.12, where y_i is the i-th true value, $\hat{y}_i$ is the predicted i-th value and $\bar{y}$ is calculated according to Equation 2.13 [43].

$$R^2 = 1 - \frac{\sum_{i=0}^{n_{samples}-1}(y_i, \hat{y}_i)^2}{\sum_{i=0}^{n_{samples}-1}(y_i - \bar{y})^2} \tag{2.12}$$

$$\bar{y} = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} y_i \tag{2.13}$$

According to the scikit-learn documentation, this score is a measure on how likely the regressor is to predict the true value [45]. One thing to note is there can be negative R^2 values, aside from the other metrics listed here, as stated in the documentation *"Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse)."* [45].

2.5.2.5 Mean Magnitude of Relative Error

The Mean Magnitude of Relative Error (MMRE), is a measure relating the prediction to the actual value [46]. The equation for MMRE is shown in Equation 2.14.

$$MMRE = \sum_{n=1}^{N} \frac{|y - \hat{y}|}{y} \tag{2.14}$$

MMRE is a relative metric not dependent on the value of the prediction. This makes MMRE applicable to comparing predictions in different data sets, apart from MSE, MAE etc. Some argue that MMRE is a measure of the spread in the prediction (standard deviation), and not the accuracy of the prediction [47].

2.5.3 Rolling origin evaluation

Evaluating time dependent data is a difficult task, as Cross validation (see Section 2.5.4) cannot be applied directly. Then the Rolling origin evaluation is an alternative [48]. The rolling origin evaluation rolls the origin of the prediction through the data set. The origin of the prediction corresponds to the place in time when the prediction is made. As this rolls forward, the frame of evaluation rolls with it, remaining the same size trough time. An example of this can be seen in Figure 2.4.

2.5.4 Cross validation

Cross validation is used for evaluating machine learning algorithms by training and testing them on a training data set [42]. Since the model has to be trained and

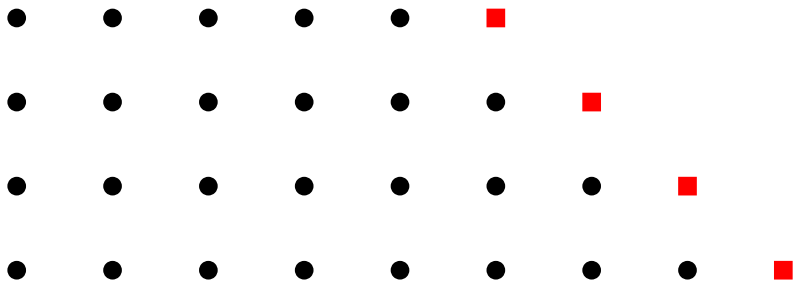

Figure 2.4: Example of rolling origin, where a red square represents the testing data, and black dots represents the training data.

evaluated on different data sets, a solution is to reserve a part of the data set for evaluation and train on the remaining data. One problem is that a model performs better when it has all the available data, cross validation helps solve this problem. K-fold cross validation is often used, were data is randomly split into K groups. Each of the K groups is then used once for validation, when all other groups are used for model building [49]. Then each model is evaluated using a scoring function and the score is then averaged over the K evaluations to reveal the overall score. The model with the best score is then used.

2.6 Key Performance Indicator

Key performance indicators (KPIs) are used today in organizations to measure a process or characteristics of products and how they change [50, 19]. As organizations strive forward in their development of services or product, measurements is required to quantify the changes as they occur. In fact, some author state that management would not exist without measurements [51]. Therefore, one suggestion is to use indicators on more aspects of an organization then just the financial results.

Dissecting the phrase Key performance indicator gives an idea of what it represents. Key, relates to the indicator being *key* to the business [52]. Performance, is defined to be measured over the future [51]. Indicator, according to the Oxford English dictionary, is something that *points out, or directs attention to, something* [53]. This can then be concluded to be a future measurement of a key aspect of an organization that indicates a change.

KPIs can often be confused with performance indicators (PIs). The difference between a KPI and PIs is thin, as PIs are not key to the business, while a KPIs are [52].

One important aspect when working with measurements is that the measurements affects the underlying behavior of the measured unit [54]. Quoting Eliyahu M. Goldratt *"Tell me how you measure me, and I will tell you how I will behave."* [55]. A fictional example of this is one software company trying to measure the productivity of developers by number of lines of code they contribute each day. This

caused problems as the productivity measures went through the roof as developers padded their code with empty lines to look more productive. Because this was the measurement of productivity, they did become more productive but this was not the desired outcome from the management.

One use case of KPIs is *The Balanced Scorecard* (BSC). BSC was formed when managers were facing problems on how to use financial indicators or operational indicators to manage an organization. Therefore one approach was to use KPIs from four areas, *financial*, *customer*, *internal* and *growth* together to govern an organization [56]. BSC does not have to apply to the top of each organization, but can also be used in sub parts of the organization as well [57].

Many characteristics of KPIs are found in literature [52, 14, 50], some are listed in Table 2.4.

Criteria	**Motivation**
Frequent Measures	Since data needs to be up to date and timely, frequent measures is a requirement for quick actions.
Acted upon	It is useless to have a KPI that is not acted upon when changes occur, then the motivation behind the measurement has to be re-evaluated.
Clearly actionable	A KPI must be actionable, as actions needs to be taken when changes occur.
Clear responsible	There must be a stakeholder responsible for the KPI and the required action.
Significant impact	If a KPI does not have a significant impact, it is not a KPI, it is a PI.
Not having too many	Having too many KPI can be confusing for stakeholders as the key focus areas is then blurred out.

Table 2.4: Some quality criteria regarding KPIs found in literature

The last point in Table 2.4 is particularly interesting, as this aspect is important when designing a KPI, as having too many KPIs can be confusing and overwhelming for the management of the organization. Having too few is also an issue, as this cannot show the complete status of the entire company in every aspect [52]. A healthy balanced is required.

2.6.1 KPIs in Software Engineering

In software engineering, KPIs play a major role in the planning and execution of developing software by giving management areas of focus. The underlying measure of a KPI is crucial for its success and to support is the ISO/IEC 15939 [5, 58, 59], established for measurements in software engineering. The ISO/IEC 15939 demonstrates the process of forming a measurement in software engineering. The process summarized is to use a *base measures* to form *derived measures* through *measurement function*. An analysis model uses the *derived measures* to form an *indicator*, the indicator is an interpretation of the metric [60]. In this context, different stakeholders are involved. According to the ISO 15939:2007 standard a

stakeholder is *"individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations"*. If the measure is not well planned and its usage well defined, chances are that it will not be utilized as it does not fulfill any of the organizations key objectives. To be a useful measurement, some quality criteria or functional criteria must be achieved. The ISO/IEC 9126 is a standardization for software quality and how it is defined. Working with these two standards, forming a KPI is now a tangible process. To explain this further an example is drawn.

2.6.2 Example

A large software company is struggling with their operation as their customers cannot reach their servers. Because of this, the CEO calls a meeting with the CTO to understand what is happening. The CTO informs the CEO that the servers are working fine when he is observing them and cannot understand what the problem is. The CEO then issues an order to create a KPI and monitor this as a result of the downtime they experienced. Since the measured entity is a aspect of ISO 9126 (Reliability), the result of this quality improvement is likely to be used. The CTO then uses the ISO 15939 to form the measurement, using data from different services monitoring the availability of the servers. The company monitors all their activity and collects as much data as possible, so this is easily accessible. This then forms a function by adding all the data sources, the function outputs the availability for all servers in minutes on a day. These minutes are then translated into a percentage of availability. The CTO goes back to the CEO for directions on what their goal is, the CEO explains that they want 99% availability. If this drops to 98%, the CTO wants immediate actions. If it is below 98%, it is really bad. The CTO then forms a traffic light model of the KPI and hands this to both CEO and the responsible stakeholders.

In this example, a number of important aspects are covered. The first important aspect is that the order came from the CEO, this indicates that the KPI is intended to be used in the balanced scorecard. Because this performance is monitored by the company CEO, the responsible stakeholders understand this and may be very keen to improve it. The second aspect is that the company measures as much as possible and have established a database of *base measures* as this makes forming new KPIs easier [61]. The third aspect is the time interval, the uptime is measured in minutes as well as updated each minute as a KPI needs to be timely [52, 14]. The frequency of update should be high to not make the KPI too static.

To continue this example, the KPI shows an availability of 95% and the CTO realizes that there is an issue with this quality. He assigns a group of engineers to work on the availability and investigate what the cause of the server failure is. The engineers discover that there is a heavy traffic load from Asia during night time and that is the reason why the servers are not performing as desired. The engineering team scales up the infrastructure to cover the server load at night and the KPI rises to 99.9% availability.

In the continuing of this example, the process of acting upon a KPI was explained. This is something that is an important part of working with KPIs [14, 52].

A famous quote by an unknown origin is *"What gets measured gets improved."*, this reflect the KPI theory well, as measurement is key.

3
Background

Before starting to work with the KPI to develop a machine learning prediction, the KPIs must be understood and evaluated. After evaluation, a KPI is chosen carefully on a number of aspects. These aspects are inspired by literature, as this reflects the current KPI theory [14].

3.1 Evaluation framework

A framework for evaluation is given in the literature [62], this approach is used when constructing new KPIs. The validation part of the framework could be applied when evaluating existing KPIs. The approach is to select an arbitrary number of measurement, so that the half of the measurements are small and the other half is large. This is presented to a reference group and they brainstorm whether the measurements are representative for the reality. To use this approach in this study, analysis of the increasing/decreasing measure is conducted and questioned whether this measure corresponds to the reality of interest.

A second more qualitative evaluation can be to apply the quality criteria listed in Table 2.4 in Chapter 2. As not all are applicable on single KPIs, the first five are selected, these are:

- Frequent
- Acted upon
- Clearly actionable
- Clear Responsibility
- Significant impact

Both of these evaluations is performed on the KPIs in this chapter.

3.2 Volvo Car Corporation KPI

The first KPI to be understood and examined is at VCC, at the software division of the propulsion department.

Here, a number of KPIs are in place, these relate to test status, development status and issue tracking. Since there is a limited time frame in this thesis work, only one KPI can be investigated. The process of selecting this was in collaboration with an industry professional which assessed all of the KPIs on usage, importance and relevance. The most important, frequently used and complete KPI related to the status of development, this was investigated further.

The KPI is used to monitor the current status and progress of the variable calibration system. As components are developed containing variables e.g. PID-regulators, there are different variables to be calibrated for optimal performance.

To keep track of these variables a database is used. This database stores the variables with a value and a status. There are 11 status levels, and each corresponds to a state of the calibration.

The calibration status is equal to a number, for the purpose of displaying an aggregated average status level. For example, if there are 5 variables in the system with the status 50 and another 5 with status 100, the average status is 75. This is what forms the KPI, as this value corresponds to a value between the lowest and highest status. Before the project is complete, a large part or all variables must have reached the highest state. As the average status changes over time, a line diagram can be displayed showing the progress over time. VCC works with gates in their development process, and at each gate a certain status level must be reached. The status levels required can be represented as a line so it is drawn on the same plot as the average value, indicating if the progress is behind, on or ahead of schedule. This can be viewed as a traffic light indicator, where above the line is equal to green, on the line is equal to yellow and below is red. Another view that is used in the KPI is the current distribution of parameters. An example of how the KPI looks is displayed in Figure 3.1 where fictional data is used.

This KPI is used along many car projects, so this knowledge can be leveraged in ML by using previous projects in predictions. As the KPI reflects the status in accurate way, and there is available data, the application of ML is a natural step.

3.2.1 Evaluation

Using the evaluations previously described in Section 3.1, this KPI can be evaluated.

An increase in this KPI corresponds to an increase in the number of variables with a higher status, or addition of high-status variables. Both of which corresponds to what the KPI is trying to capture.

A decrease in this KPI can indicate a degradation of some status variables, or an addition of low-status variables that dilute the average level. These two cases is both representative for the KPI, as both of these indicate an additional work load on the teams.

There is also some noise of unwanted behavior in the data e.g. when an variable changes name it status is automatically 0 again.

The five quality criteria are answered below:

- Because the KPI is measured on a weekly basis, updates in the underlying variables can be seen the next week.
- The stakeholder responsible for this KPI must follow a plan, and if the status does not comply with the plan action has to be taken.
- The clear action to be taken in the case of falling behind schedule is to increase the development effort to raise the average status. This has to be done carefully as the actual status has to be raised only when the variable reaches a level of maturity to not compromise the accuracy of the KPI.
- The responsible stakeholder for this KPI is clear as this is related to the

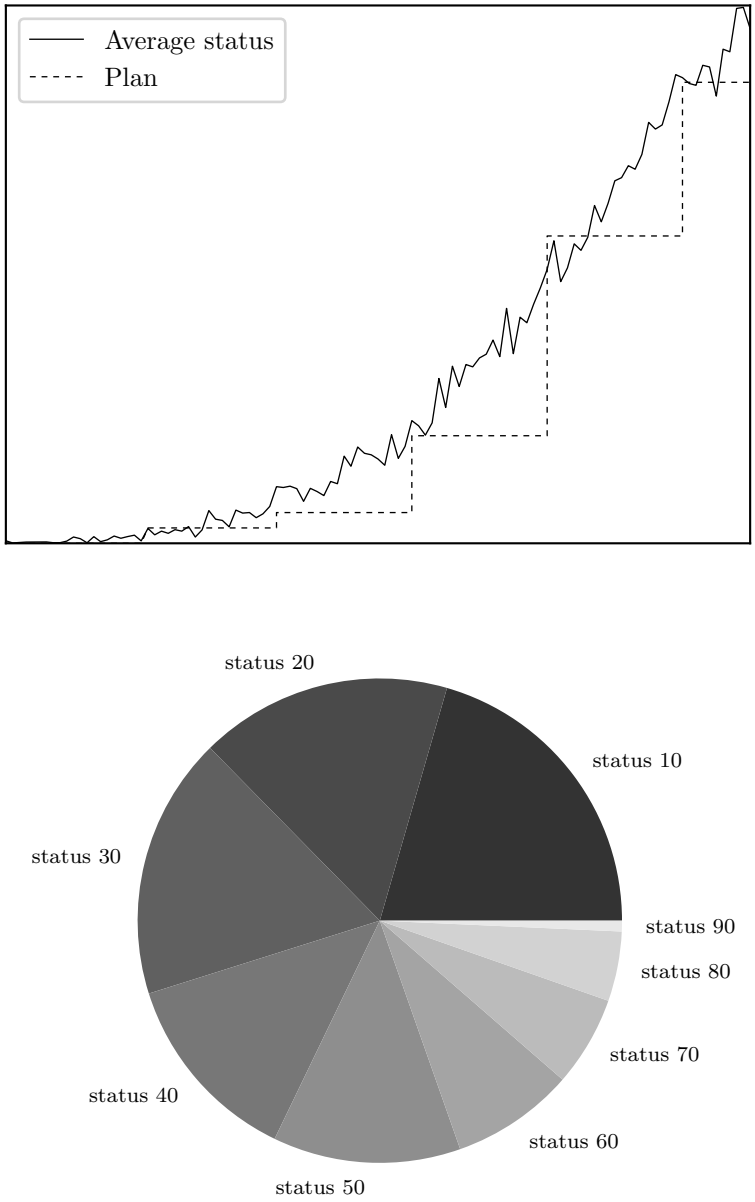

Figure 3.1: Example of how the status KPI can look

status level of parameters. As the KPI displays a second view apart from the average status, by the distribution of parameters via pie-charts, this makes it clear where in the process the bottle neck is. As some parts are larger then others, this indicates where the majority of the work is required.
- The significant impact of this KPI can be understood by the involvement of parties, as software developers, calibrators and managers are affected by this KPI it stretches over the entire process of implementing components in a car.

3.3 Ericsson AB KPI

At Ericsson AB, the usage of KPIs in software is immense. The investigation aimed to target two of the most known KPIs. The KPIs selected regarded the inflow of defects as well as the backlog of defects and how this changes over time. This is interesting to study as accurate predictions in this area could help to e.g. to have an overview of the releases that are about to be released on the market soon.

The inflow relates to the number of defects reported to a defect system, and the backlog refers to the number of defects not yet closed in the defect database.

A defect, in this context, is a bug within a code segment or a design flaw. In this study internal defects are studied, that is defects reported during development. The defect database consists of numerous information e.g. the discovery process of a defect as well as documented time stamps of each stage of a defect (Reported, inspected and fixed). For this report a defect is considered reported when it first appears in the defect system.

The acting upon these KPIs are done by monitoring the trend of the data. If the trend indicates that the number of defects are increasing, then e.g. more resources may need to be allocated to fix the defects. If the trend of defects indicates a decline, no action is taken.

The KPI of defect inflow can indicate how effective the organization is in finding defects. If the amount of incoming defects decreases, this may e.g. indicate that the organization is occupied with fixing existing defects, or that the development of the new software is coming to an end (i.e. will soon be released to the marked).

The backlog KPI indicates the strain on the organization with its ability to manage the in- and outflow efficiently [63].

3.3.1 Evaluation

Using the evaluation previously described in Section 3.1 these two KPIs can be evaluated.

If the reporting of defects decreases while the effort spend finding defects remain the same, the number of defects not found in the software could be considered as decreasing.

An increase in the backlog indicates that the inflow of defects is greater than the outflow and a decrease indicates a higher outflow then inflow. If the resources spend on locating and fixing defects is equal, while the backlog increases and more resources are required or overtime is needed.

A more qualitative evaluation can be seen below:

- These KPIs is updated on a daily basis it can be considered as frequent.
- The action on these KPIs, as previously discussed, is by increasing the number of developers fixing defects or shift focus of developers to fixing defects.
- An action to take when the backlog grows too large is to pivot the focus to fixing defects, but as the focus is pivoted the number of incoming defects will decrease which may affect the inflow measure e.g. less defects may be reported. During this period the focus must lie on the backlog KPI.
- The clear responsibility is the team leaders (or other leading roles) governing the development/defect fixing focus.
- The significant impact of the correct use of these two KPIs yields an efficient handling of defects, which also yields faster development of software.

4
Method

When the KPI is understood and evaluated the second step is to gather available data, as if the amount of data is not sufficient, generalization is difficult. Applying ML to KPIs is an iterative process and each iteration has new insights. This chapter explains the process of finding a suitable ML algorithm and how to format the input data to fit this algorithm.

The framework used for applying the ML algorithms is Scikit-learn [39] which is a complete library for creating models, train them and measure the prediction. The version of Scikit-learn used in this report is 0.18.

4.1 Overview

In this thesis, six research cycles was carried out. Cycle 1 to 4 investigates the VCC status KPIs and cycle 5 to 6 investigate the KPIs at Ericsson AB.

The first cycle applies ML to the average status, this is then changed in the second cycle to investigate the distribution of parameters in the status KPI. The third cycles applies the rolling time frame to the same data set as in the first cycle (average status) and the fourth cycle applies the rolling time frame to the same data set as the second cycle (distribution of parameters).

The fifth cycle is carried out at Ericsson AB with the use of a defect inflow KPI and the rolling time frame. The sixth and last cycle applies the rolling time frame to the defect backlog KPI at Ericsson AB.

The motivation for using action research is to being able to iterate over the process and test different solutions in a way that is easy to trace.

4.2 Machine learning process

The process of finding a suitable machine learning algorithm is a straight forward process which includes the following steps [49, 3]:

1. Gather raw data.
2. Split the data into testing and training sets.
3. Clean the data and transform it in to a correct representation that the Machine Learning algorithm can use.
4. Find a suitable algorithm e.g. by using Figure 2.3.
5. Run the algorithm with hyper parameters.
6. Evaluate the performance of the algorithm.
7. Optimize the hyper parameters or change algorithm.

This process can be depicted in Figure 4.1.

This was the strategy used in this report. Machine learning includes a lot of trial and error (since it is often not obvious if there is a underlying pattern to be discovered or not), this report will be divided into different cycles.

A lot of this process can be automated, either by using functions that are established by the framework, Scikit-learn, or by writing custom methods. This leads to a lot of information about results that are not required to be disclosed and will therefore not be presented in the results section. An example of this is the *GridSearchCV* method in the Scikit-learn library that establishes a grid of combinations of parameters to apply to cross validation and select the best model thereafter.

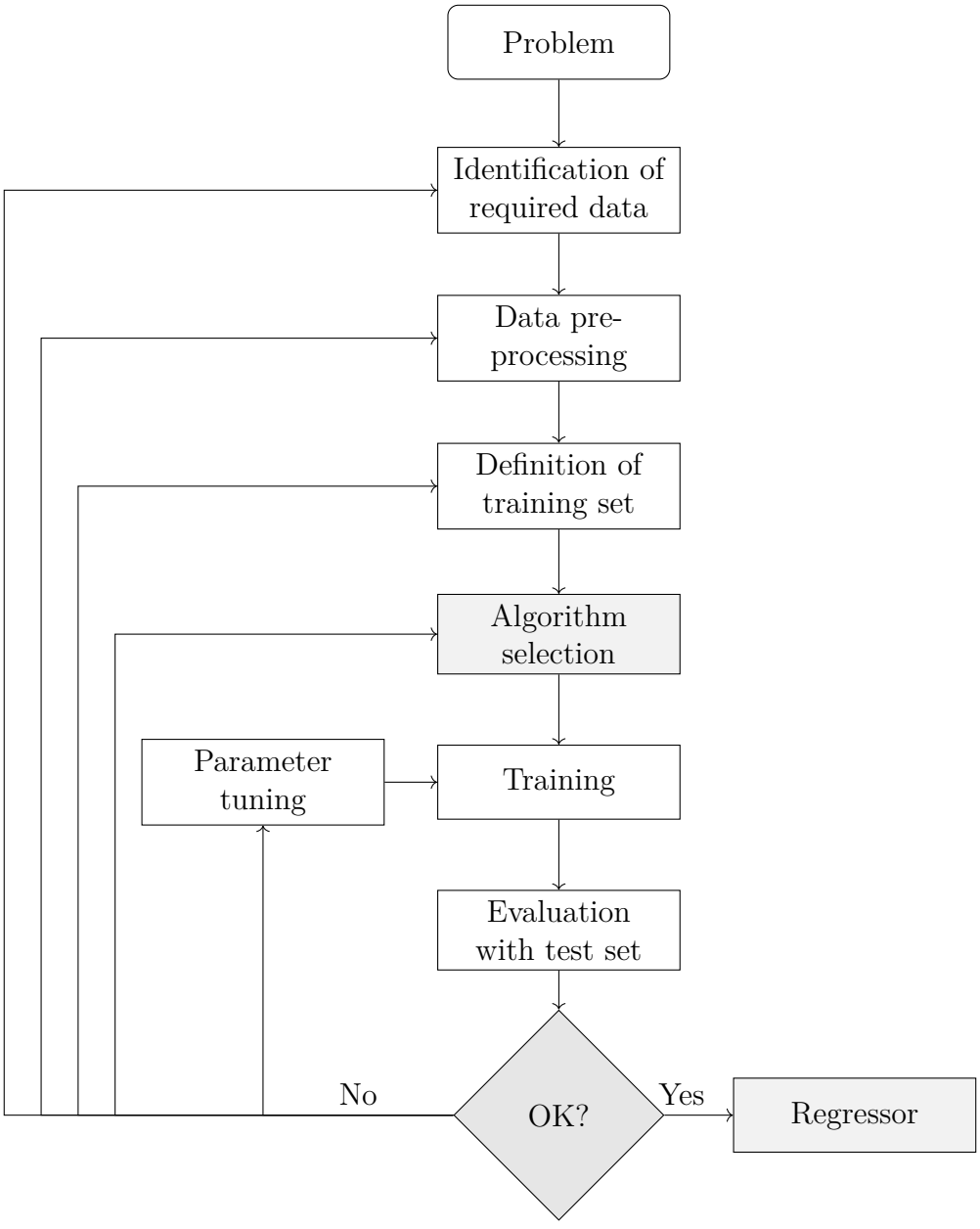

Figure 4.1: Process of finding a ML regressor from [3]

4.3 Action Research Cycle 1

The first cycle was made at VCC, using their current KPI with status of software parameters. Here, the objective was to predict the average status of the project using previous projects. The first cycle included gather information on how KPIs are acted on and what raw data is available.

4.3.1 Data gathering

The raw data available was the underlying data used to form their current KPIs. This data was even more detailed then the KPI, as the KPI aggregated the data.

Data aggregated to some level, was extracted from a database. The data contained information about the supplier for each component, their placement, the module name, the related car project, and how many parameters each status bin contained. This was arranged by weeks and there were approximately 20,000 rows of data. An example of a data row is seen in Table 4.1, where the headings **10...120** refers to the status bins of all 11 status levels. In the status bin, the number of parameters labeled with this status is displayed. An average value was then calculated for each week using the bins and number of parameters.

Supplier	**Placement**	**Module name**	**Project**	**Week**	**10**	...	**120**
Supplier A	Front	Module ABC	Project 1	15w04	4	...	1

Table 4.1: Part of example data from VCC

4.3.2 Splitting data into training and testing sets

When splitting data into training and testing sets there are a few options. A commonly used method is cross-validation [23], but this cannot be applied to time dependent data [42]. Since the cross-validation divides the data at random and tests and trains on different splits, this would break the dependence in time-dependent data. Instead a different split is used, this is done by time frames of $Length/Splits = Chunk$, where $Length$ is the length of the data and $Split$ is the number of splits, the length of the testing data is always the same ($Chunk$), but the size of the training data grows ($i * Chunk$, where i is the i-th split). This is referred to as a rolling origin evaluation. This was to simulate a real situation where time has elapsed and data has been gathered and is now used in predictions. Then after a certain time period the prediction is evaluated to measure its accuracy. The number of splits to be done had to be based on how long a prediction would be. If a longer time was desired, training a regressor on smaller time frames is not useful.

A split of 90% testing data and 10% training data will be used in this cycle. As there are previous projects, these will be used together with the 90% training data to form a model and predict the last 10% which will be compared to the true values.

4.3.3 Data cleaning

The data had to be formatted correctly to fit the ML algorithm. Since VCC format their time stamps using a week number and a year, this was the first task to format.

The data was then grouped by week and project, and the average status for each week was measured. This was the data used in this prediction.

VCC separates data by projects, a one hot encoding was performed on the projects. Data from some of the projects were not in place from the project start, so they were removed from the data set. This decision was made because the data was not relevant and did not provide any extra information.

The project start week was then normalized to make all projects start at the same date to make the week correspond to the week of a project. This change were discussed with an experienced stakeholder who emphasized that the project week has more characteristics then the week of the year.

One of the projects was about half the size then all other projects, so the week number of this project was doubled to make the data more similar to the other projects. The reason for multiplying the week number by two was tested to determine the factor, two was a good approximation graphically when plotting the average status. The drawbacks of this was tampering with the data could imply that more customization would be required before accurate predictions could be made on future projects. As car projects at VCC can be of different magnitude, some are complete redesigns while others are just a yearly update. This made it obvious that size has to be taken into account when doing predictions on new projects. If as stakeholder could estimate the size of the new project to be similar to that of a previous project, then predictions will be more accurate.

4.3.4 Finding an algorithm

To find an algorithm, Figure 2.3 was followed. There is more then 50 samples and a category is not predicted, the use of a regressor was established. Since there were 191 data points in the data frame at this point, the *SGDRegressor* was discarded as it requires more then 100 000 samples. Because there was one feature (Week) Lasso regression and elastic net were discarded. This lead to the use of Ridge regression, Linear SVR and SVR with the RBF kernel. As computation is not too expensive as the data set is small SVR with the polynomial kernel was also included as the curves resembled a polynomial function.

4.3.5 Evaluating an algorithm

To find the best suitable algorithm, a metric for measuring the fitness or accuracy of a prediction is needed.An indication on how close to the actual value is to the prediction, is required. There are a number of possible evaluators: mean squared error (MSE), mean absolute error (MAE), median absolute error (MedAE) and R^2-score. Some authors suggest to use MAE as this is not as sensitive to high variance in the error, but this is not desired in this case [4], so MSE is used.

4.3.6 Iteration

When the evaluation is finished, the score of the regressor is tested against the best regressor. This process was done over an array of regressors created by iterating over different ranges and combination of hyperparameters. After the iteration has terminated, the best score and regressor was saved. The hyper parameters is presented in Table 4.2, the ranges was originally larger, but was shrunk as they seemed too large and the entire project took too long to execute.

Algorithm	Parameter	Range
SVR	kernel	$[RBF, Poly]$
	C	$10^x, x \in [-5, -3)$
	epsilon	$10^x \in [-1, 2)$
	gamma	$[0.1, 0.5)$
SVR linear kernel	C	$10^x, x \in [-2, 1)$
	epsilon	$10^x, x \in [-2, 2)$
	intercept scaling	$[0.8, 1.0)$
Ridge regression	alpha	$10^x, x \in [-5, 5)$
	normalize	$[True, False]$
	tol	$10^x, x \in [-5, 5)$

Table 4.2: Hyper parameters used in the first cycle

To clarify Table 4.2, the linear SVR kernel has no gamma value. The intercept scaling in the linear kernel is only available for this optimized version of the linear kernel. The tol variable is a stopping criteria for the accepted error for the ridge regression to reach a solution.

4.4 Action Research Cycle 2

The second cycle was also performed at VCC and aimed towards predicting the number of parameters in each status bin. This meant predicting the number of parameters belonging to a certain status at a given time. The knowledge of how variables changes over time is useful to understand as this can provide insight toward more accurate planning. Since many stakeholders is involved in the development/-configuration process, knowledge about how many parameters reaches their level of responsibility each week can help stakeholders prepare and plan.

4.4.1 Data gathering

The data gathering was very similar to the previous cycle without average calculation.

4.4.2 Splitting data into training and testing sets

This cycle used the exact time split as the previous cycle.

4.4.3 Data cleaning

This was also very similar to the previous cycle, the week number of the smaller project was doubled. The week numbers were also normalized to start at zero, as in cycle one.

4.4.4 Finding an algorithm

As no new features was added, this cycle came to the exact same conclusion as the previous cycle.

Since the labels to predict is an array of $L*11$ features, where L is the length of the prediction, a few changes has to be made compared to predicting a one dimensional array. Since this multidimensional output is not as common, the regressors lacking support of these has to be customized to fit this purpose. This can be done using the Scikit-learn function *MultiOutputRegressor*. This function takes a regressor only able to predict one dimension, and the input data and labels, then produces X regressors, where X is the width of the output data. For example, in this case were it is an output width of 11. The function then creates 11 regressors, one for each label, then predicts with the 11 regressors separately and join the result to a multidimensional array.

4.4.5 Evaluating an algorithm

This cycle evaluated a multidimensional output, so some additions to the previous equation was necessary. The multidimensional output was evaluated bin by bin, then the average status of all bins were then averaged in the end of the computation.

4.4.6 Iteration

The iteration was similar to the first cycle, in splitting and creating regressors. The only difference this time there were more data as the number of parameters in each bin was used. The hyper parameter used can be seen in Table 4.3.

Algorithm	**Parameter**	**Range**
SVR	kernel	$[RBF, Poly]$
	C	$10^x, x \in [-4, -4)$
	epsilon	$10^x \in [-3, 3)$
	gamma	$10^x \in [-4, 4)$
SVR linear kernel	C	$10^x, x \in [-1, 1)$
	epsilon	$10^x, x \in [-1, 1)$
	intercept scaling	$[0.9, 1.2)$
Ridge regression	alpha	$10^x, x \in [-1, 1)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	tol	$10^x, x \in [-5, 2)$

Table 4.3: Hyper parameters used in the second research cycle

4.5 Action research cycle 3

This research cycle was also performed at VCC, but with a different approach to data modeling. The same data calculated in research cycle 1 was again evaluated.

4.5.1 Gathering data

The third cycle used the same data that was gathered in cycle 1, the average status each week.

4.5.2 Splitting data into training and testing sets

The splitting of train and test data was performed by using 10% testing data and 90% training data, same as the previous cycle.

4.5.3 Data cleaning

There is a time dependency in the data and had to be modeled accordingly. The problem was that a ML algorithm cannot account for the order of the data set and use this ordering to predict future values. This is solved by the way the data was modeled, by capturing the time dependency. This was done by using a *rolling window*.

4.5.4 Finding an algorithm

An inducer algorithm, such as Linear Regression or a SVR with a linear kernel, models the data in a completely different way than an instance based learner such as KNN. Selecting one over the other can yield different results. Since there is a difference in implementation between a Linear SVR and a Linear Regression algorithm, both of these are investigated, together with KNN. As some sources argue that the RBF kernel of the SVR is also seen as an instanced based learner, this is also included [49]. Since Linear Regression has been extended by Lasso, Ridge and Elastic Net, these are also included for comparison.

Since this data is not modeled as the previous cycle, cross validation can be used to valuate these models. The testing data set is set to 10% of the total data, a 9 fold cross validation is applied to make each fold the same size as the testing data. Scikit-learn implements a function named *GridSearchCV*, which is short for *grid search cross validation*, this can be used to determine the optimal algorithm together with a number of specified hyperparameters. GridSearchCV establishes a grid of each algorithm with all possible combination of the parameters given and their respective values and by utilizing cross validation and the R^2 score finds the optimal solution.

4.5.5 Evaluation of algorithm

There are five different algorithms to measure, and they all use the X_{t-1} observation in prediction, one strategy to evaluate this is by introducing a dummy estimator.

The dummy estimator predicts the previous days value, regardless of the other previous values. This is equivalent with having a linear estimator with the $\beta_0 = 0, \beta_1 = 1, \beta_2 = 0, ..., \beta_n = 0$. If the prediction cannot achieve a better prediction then this then the predictions are useless.

To find the optimal algorithm, different lag values had to be investigated. In this cycle lag from one up to ten was evaluated. The range for different hyper parameters used for each model can be seen in Table 4.4.

Algorithm	**Parameter**	**Range**
KNN	neighbors	$[2, 20)$
	weights	$[uniform, distance]$
	algorithm	$[auto, ball_tree, kd_tree, brute]$
	p	$[2, 20)$
SVR Linear	C	$[0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10, 50, 100, 500]$
	epsilon	$[0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500]$
	loss	$[epsilon_insensitive, squared_epsilon_insensitive]$
Elastic Net	alpha	$10^x, x \in [-3, 3)$
	l1 ratio	[0.1..1.0]
	normalize	$[True, False]$
Lasso	alpha	$10^x, x \in [-3, 3)$
	normalize	$[True, False]$
Ridge	alpha	$10^x, x \in [-3, 3)$
	normalize	$[True, False]$
SVR RBF	C	$10^x, x \in [-2, 1)$
	epsilon	$10^x, x \in [-3, 4)$
Linear regression	fit intercept	$[True, False]$
	normalize	$[True, False]$

Table 4.4: Hyper parameters used in the third cycle

4.5.6 Optimization

Because cross validation is used, the only optimization needed is to explore the different hyper parameters. The hyper parameters tune the algorithm to the optimal performance. The cross validation uses the R^2 score internally to investigate the best model.

To be able to compare the results from the previous cycle the result is reported in a MMRE score.

4.6 Action Research cycle 4

In cycle four the same KPI as previous was investigated at VCC, in this cycle the status bins of the parameters was investigated. This cycle applied the same model as the third cycle with a time lag.

4.6.1 Data gathering

As this cycle used the same data as cycle 2, the process of data collection is the same.

4.6.2 Data cleaning

In this cycle, the 11 status levels were predicted with the time lag of each status level. This means $lag * 11$ different features to predict the 11 status levels after the rolling window is applied. Each status level was lagged a number of weeks to create the features for the prediction.

4.6.3 Finding algorithms

The algorithms to use was either instance based or an inducer. To cover as many of these as possible, KNN, Lasso, Elastic Net, Linear SVR, RBF SVR and Linear Regression is evaluated. This is to cover as many of each as possible and find the most suitable one.

In this cycle, as in the previous one, lags from one to ten was investigated.

4.6.4 Evaluation of algorithms

Like in the previous cycle where cross validation is applied, here the evaluation is done using a R^2-score. In this case, the dummy estimator was also used to show the performance of the predictions.

The result of these predictions are then averaged to the same formulation as the KPI, and the result of this prediction is reported using a MMRE score. This is to be able to compare the results of these predictions with the previous cycles.

The hyper parameters used in the fifth cycle is shown in Table 4.5.

4.7 Action Research Cycle 5

In this cycle the KPI from Ericsson AB was investigated. The KPI was regarding the inflow of defects during development of a single product at Ericsson AB.

4.7.1 Data gathering

The data was structured in databases regarding different products, but as this cycle only set out to investigate one product the need for multiple databases was unnecessary. The script used a dump of data from the database to provide the same data each run to not compromise the validity of the predictions between algorithms.

The defect database contained all defects with their respective name and time stamp, as well as if it was a duplicate of another defect. The duplicate defects were removed. The lasting defects regarding the development phase of a single product were selected. These were then grouped by the incoming date on a weekly basis and counted to result in the number of defects added each week.

Algorithm	Parameter	Range
KNN	neighbors	$[2, 30)$
	weights	$[uniform, distance]$
	algorithm	$[auto, ball_tree, kd_tree, brute]$
	p	$[2, 10)$
SVR Linear	C	$10^x, x \in [-2, 3)$
	epsilon	$10^x, x \in [-3, 2)$
	dual	$[True, False]$
	loss	$[squared_epsilon_insensitive]$
Elastic Net	alpha	$10^x, x \in [-3, 3)$
	l1 ratio	[0.1..1.0]
	fit intercept	$[True, False]$
	normalize	$[True, False]$
Lasso	alpha	$10^x, x \in [-3, 3)$
	fit intercept	$[True, False]$
	normalize	$[True, False]$
Ridge	alpha	$10^x, x \in [-3, 3)$
	normalize	$[True, False]$
SVR RBF	C	$10^x, x \in [-2, 3)$
	epsilon	$10^x, x \in [-3, 4)$
Linear regression	fit intercept	$[True, False]$
	normalize	$[True, False]$

Table 4.5: Hyper parameters used in the fourth cycle

4.7.2 Data cleaning

Similar to the previous KPIs at VCC this KPI is time dependent, therefore the use of the same rolling window approach was used. As stakeholders at Ericsson AB stressed, the need for predictions longer than one week was not desired.

4.7.3 Finding algorithms

Similar to previous cycles, the preference of a instance based learner or a inducer was still unknown, therefore both types were evaluated. The models evaluated was: Linear SVR, Linear Regression, Lasso, Ridge, KNN and Elastic Net.

Since cross validation was applied, and hyper parameters is specified in ranges, one important key to investigate if any of the hyper parameters had reached the end of the range. This can indicate that further improvements can be made as increasing or decreasing a variable can impact the performance. Since the range of hyper parameters should not grow too large, a logarithmic scale was used to determine the magnitude of the variable (e.g.: $10^x, x \in [-4, 4]$). When the magnitude was decided, a closer approximation to the optimal value could be made. A table of the hyper parameters used can be found in Table 4.6.

Algorithm	Parameter	Range
SVR Linear	C	$10^x, x \in [-4, 4)$
	epsilon	$10^x, x \in [-4, 4)$
Linear Regression	normalize	$[True, False]$
	fit intercept	$[True, False]$
Lasso	alpha	$10^x, \in [-5, 5)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	tol	$10^x, x \in [-10, 2)$
Ridge	alpha	$10^x, x \in [-5, 10)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	solver	$[auto, svd, cholesky, lsqr, sparse_cg, sag]$
KNN	N neighbors	$[2, 30)$
	weights	$[uniform, distance]$
	p	$[1, 10)$
Elastic Net	alpha	$10^x, x \in [-3, 3)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	tol	$10^x, x \in [-3, 1)$
	l1 ratio	[0.0...1.1]

Table 4.6: Hyper parameters used in fifth cycle, calculating the defect inflow

4.7.4 Evaluation of algorithms

Like in the previous cycle where cross validation was applied, the internal evaluation was done using a R^2-score. To compare different models and different lags, the MMRE score was used.

4.8 Action Research Cycle 6

In this cycle the KPI from Ericsson AB was investigated. The KPI was regarding the backlog of defects of a single product during development at Ericsson AB.

4.8.1 Data gathering

This data was structured in a similar way to the previous data, so the backlog had to be counted manually. Where the delta of the grouped in- and outflow was added to a value per week to form the number of defects active in the backlog. As with the previous cycle, duplicates were removed.

4.8.2 Data cleaning

The backlog was time dependent, so it was modeled in a similar way to previous cycles, using a rolling window. As this KPI was used in junction with the previous

Algorithm	Parameter	Range
SVR Linear	C	$10^x, x \in [-4, 4)$
	epsilon	$10^x, x \in [-4, 4)$
Linear Regression	normalize	$[True, False]$
	fit intercept	$[True, False]$
Lasso	alpha	$10^x, \in [-5, 5)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	tol	$10^x, x \in [-5, -2)$
Ridge	alpha	$10^x, x \in [-5, 10)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	solver	$[auto, svd, cholesky, lsqr, sparse_cg, sag]$
KNN	N neighbors	$[3, 12)$
	algorithm	$[ball_tree, kd_tree, brute, auto]$
	p	$[1, 10)$
Elastic Net	alpha	$10^x, x \in [-3, 3)$
	normalize	$[True, False]$
	fit intercept	$[True, False]$
	l1 ratio	[0.0...1.1]

Table 4.7: Hyper parameters used in sixth cycle, calculating the defect backlog

one (inflow of defects) measuring more than a week ahead was not useful.

4.8.3 Finding algorithms

Again, the need for an instance based learner or inducer was unknown so the same models as the previous cycle were used. The hyper parameters can be found in Table 4.7.

4.8.4 Evaluation of algorithms

Like in the previous cycles where cross validation was applied, the internal evaluation was done using a R^2-score. Externally, to compare different lags and models, MMRE was used.

5
Results

In this chapter, the results from the experiments will be presented. To be able to interpret the output of the algorithm, the data is aggregated to an average level, by project, week and average value. This is then used to visualize the prediction. Because of confidentiality reasons, the graphs will have stripped axes and MMRE will be used to evaluate the predictions.

5.1 Action Research cycle 1

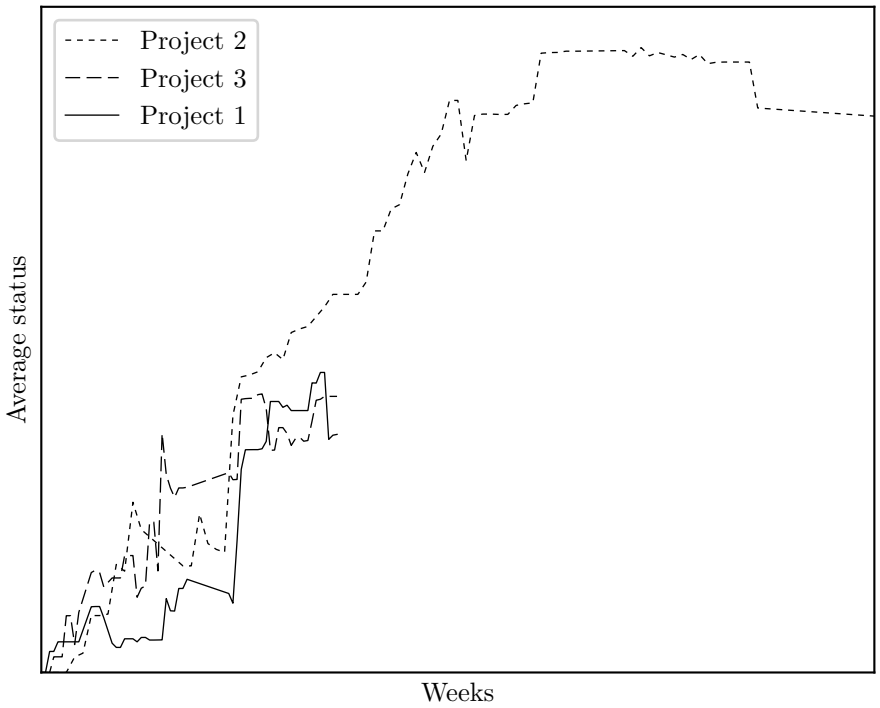

Figure 5.1: Plot of the average of projects before feeding the algorithm

Before presenting the result from the first cycle, the data used is presented in Figure 5.1, here the x-axis is weeks and the y-axis is the average status of each project.

The result from the first cycle can be seen in Figure 5.2. The dotted line is used together with the previous projects (project 2 and 3 in Figure 5.1) to form

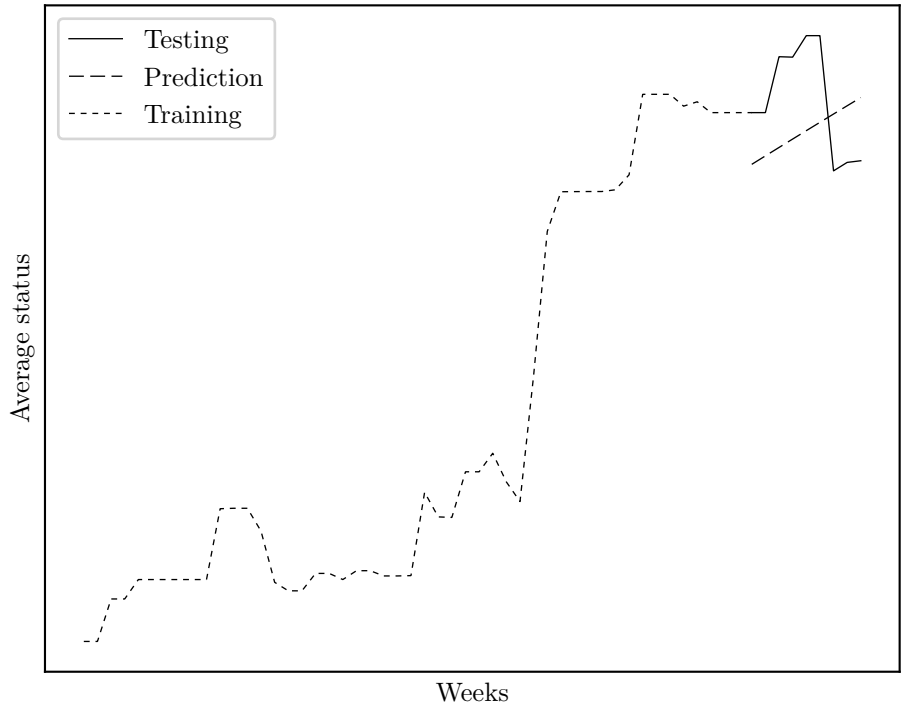

Figure 5.2: Result from first cycle

the model. The model predicts the dashed line given the week number of the solid testing line. The MMRE of this prediction is **0.1273**, the data used for evaluation was the last 10% of the data set.

Figure 5.3 displays an enhanced graph of the prediction.

The prediction was made using a support vector regressor with a polynomial kernel, the specifications can be seen in Talbe 5.1.

Parameter	**Value**
C	0.0001
coef0	0
degree	3
epsilon	10
gamma	0.3
kernel	poly
shrinking	True
tol	0.001

Table 5.1: Parameters used for best model in first research cycle

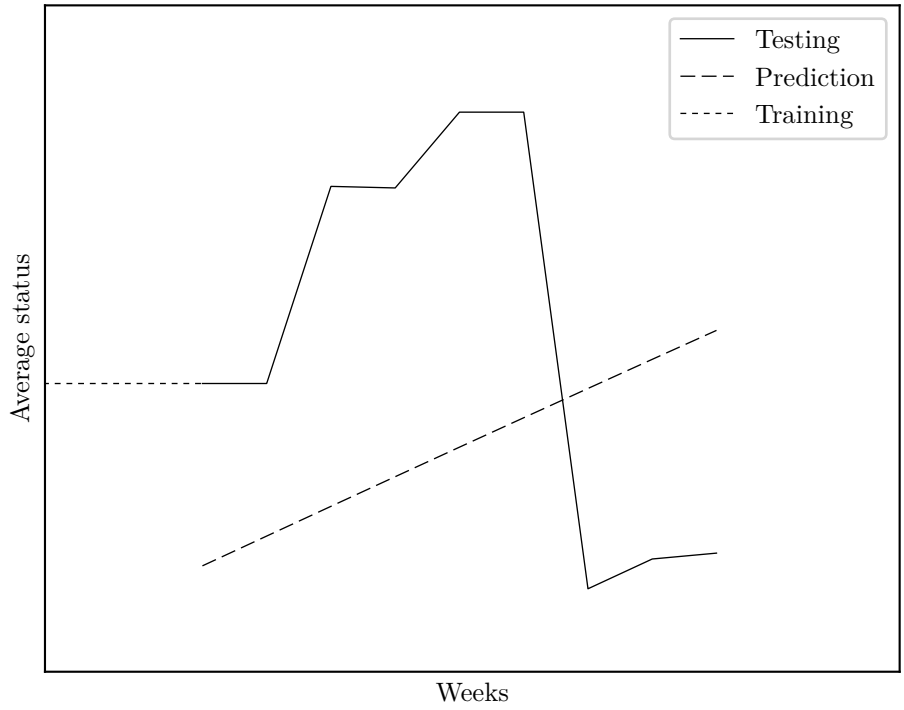

Figure 5.3: Enhanced result from first cycle

5.2 Action Research cycle 2

The second cycle used a multidimensional output with the same input as the previous cycle.

The result from the second cycle was aggregated to a weekly level, using the average status of the prediction, to be able to compare to the previous cycle.

In Figure 5.4, the result from the prediction can be seen. Here, the dotted line is the training data, together with previous projects (project 2 and 3 in Figure 5.1), the dashed line the prediction and the solid line the actual result. The last 10% of the data set was used for evaluation and resulted in a MMRE of **0.2047**.

In Figure 5.5, an upscale version is displayed. The predictions were made using a support vector regressor with the RBF kernel, the parameters used can be seen in Table 5.2. As SVRs cannot predict multidimensional outputs, a helper function was used.

In Figure 5.6 the distribution of the status levels for project 1 can be shown, and how these changes over time. The x-axis corresponds to the number of parameters included in each status level. In this graph it is clear that status level 85 is not used more than once, and level 110 is not used at all.

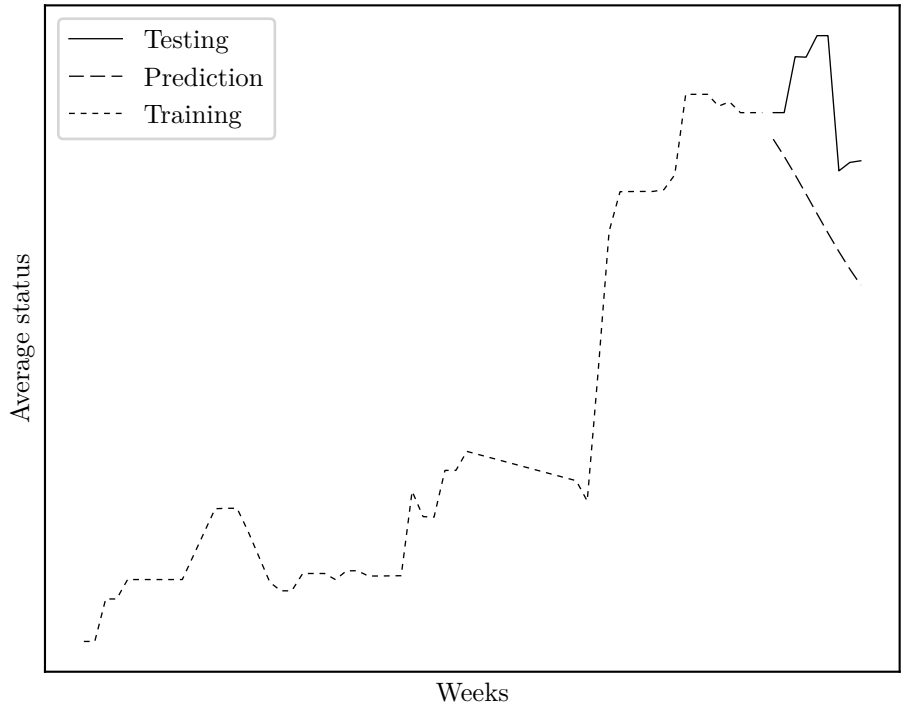

Figure 5.4: Result from the second cycle

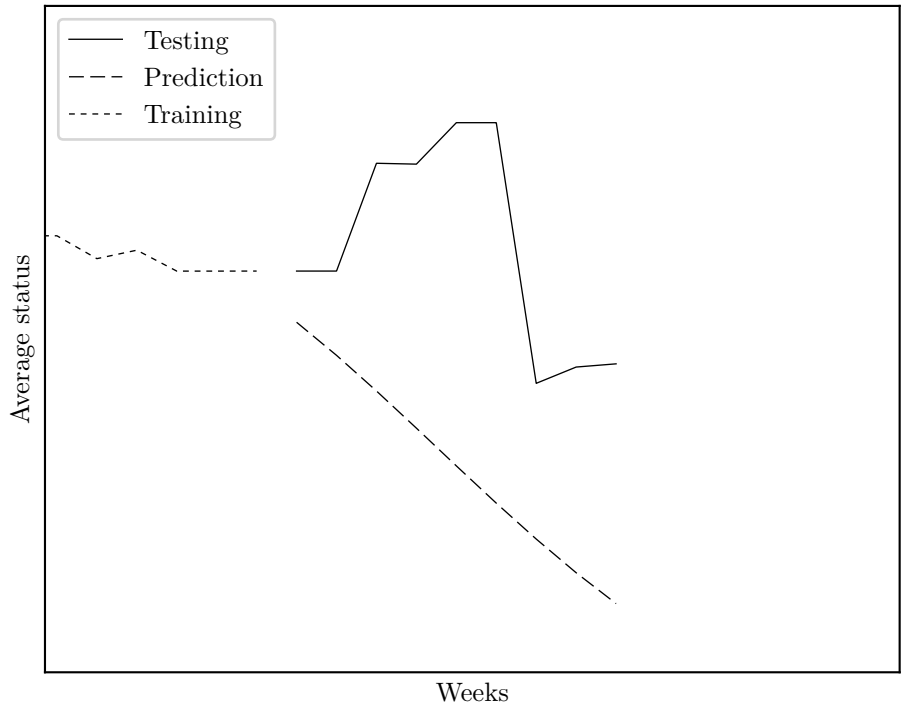

Figure 5.5: Enhanced result from cycle 2.

Parameter	Value
C	1000
coef0	0
degree	3
epsilon	100
gamma	0.01
kernel	rbf
shrinking	True
tol	0.001

Table 5.2: Parameters used for best model in second research cycle

5.3 Action Research cycle 3

The third cycle was performed at VCC using the rolling window approach on the time series. To compare the prediction, a benchmarking score is used by returning the previously observed value. The result from the third cycle is displayed in Table 5.3, where the value in bold is the best prediction and the italic value is the best non-benchmark prediction. The best score achieved was by the benchmarking measure and tangents the Linear SVR score, therefore no plot is displayed for the prediction of this model.

The division here was to use 80% as training data and 20% as testing data. The reason for the benchmarked value not remaining the same is the rolling window was calculated after splitting the data, and the rolling window removes as many data points from the data set as the length of the lag. Since the data set to test on was fairly small this change is visible.

lag	Bench-mark	KNN	Linear SVR	Elastic Net	Lasso	Ridge	RBF SVR	Linear Regress-ion
1	0.0400	0.0893	0.0574	0.0723	0.0723	0.0723	0.1887	0.0438
2	0.0440	0.0854	0.0608	0.0742	0.0742	0.0742	0.1887	0.0487
3	0.0489	0.1153	0.0574	0.0827	0.0827	0.0827	0.2182	0.0548
4	0.0550	0.1220	0.0580	0.0836	0.0836	0.0836	0.2255	0.0604
5	0.0629	0.1271	0.0596	0.0833	0.0833	0.0833	0.2364	0.0716
6	0.0574	0.1273	0.0589	0.0820	0.0820	0.0820	0.2364	0.0630
7	0.0688	0.1098	0.0607	0.0830	0.0830	0.0817	0.2364	0.0722
8	0.0771	0.0770	0.0611	0.0850	0.0850	0.0836	0.2364	0.0823
9	0.1028	0.1054	0.0769	0.0719	0.0719	0.0719	0.2364	0.1099
10	**0.0105**	0.1061	0.0653	0.0756	0.0756	0.0750	0.2364	*0.0125*

Table 5.3: Result from cycle 3

This result shows that the best prediction to recommend is using the previous value to estimate a new value. One reason behind this result is the strong relationship between values as this yields a low error when the benchmarking function

is used, no model can outperform this estimate. One conclusion to be drawn is that the inducer algorithms outperform the instance based algorithms. This can be interpreted as there is a strong correlation between the values in the time series.

5.4 Action Research cycle 4

The fourth cycle was also performed at VCC, but predicting a multidimensional output instead. Here the values were averaged to the same level as when used as a KPI. This was to be able to compute the MMRE and compare against the previous result. Here, the benchmark score is the same as in the previous cycle, since the average value of a previous time step is equal to the calculated average value of the same time step. The result of the prediction can be seen in Table 5.4, the value marked in bold is the best prediction made and the italic value is the best non-benchmark value.

lag	Bench-mark	KNN	Linear SVR	Elastic Net	Lasso	Ridge	RBF SVR	Linear Regress-ion
1	0.0400	0.1030	0.0792	0.0827	*0.0426*	0.0854	0.4432	0.0796
2	0.0440	0.1344	0.1831	0.0449	0.0449	0.0807	0.4804	0.0865
3	0.0489	0.1279	0.1289	0.0510	0.0510	0.0840	0.5004	0.0741
4	0.0550	0.2032	0.1678	0.0568	0.0568	0.1260	0.5153	0.1169
5	0.0629	0.2318	0.2147	0.0648	0.0648	0.2472	0.5514	0.1915
6	0.0574	0.2403	0.4904	0.0634	0.0634	0.3552	0.6213	0.8202
7	0.0688	0.1856	1.6440	0.0799	0.0799	0.4527	0.7155	1.0761
8	0.0771	0.1206	1.6794	0.1620	0.1620	0.5825	0.8457	0.7294
9	0.1028	0.1090	0.6563	0.1991	0.1991	0.7710	0.9703	15.4470
10	**0.0105**	0.2718	0.9400	0.0454	0.7852	0.0454	0.9654	4.2518

Table 5.4: Result from cycle 4

The prediction of this round could not outperform the benchmark, leaving the predictions unsatisfactory. An interesting result is the resemblance between the Elastic net and the Lasso, as these have the exact same error the prediction is most likely to be equivalent. The explanation for this is the formulation for the Elastic net, as this can have the exact same form as a Lasso with the right hyper parameters.

The better algorithms in this case is the Lasso and Elastic net, as these have a relatively low error throughout. In contrast to the previous cycle, there is no clear preference between inducer or instance based learner.

5.5 Action Research cycle 5

In this cycle the KPI regarding the defect inflow at Ericsson AB was investigated. The available data can be seen in Figure 5.7. This graph shows the number of defect that has been reported each week. For confidentiality reasons, the axis has been erased.

In Table 5.5, the result from the cycle is listed. This measure was done using 90% of the available data and predicting the last 10%, this also applies to the plotting of the predictions. The best model was a KNN algorithm with a lag of 8. The KNN used the closes 28 neighbors to predict this result. As this result is better than the benchmark prediction, a MMRE of 0.1925 is still rather high as this indicated that the prediction had an error of almost 20%. A trend in the data is that the instance based learner outperforms the inducer algorithms. The difference is not substantial but it is a steady trend in the result.

Lag	Bench-mark	Linear SVR	Linear Regression	Lasso	Ridge	KNN	Elastic Net
1	0.2707	0.2295	0.2489	0.2489	0.2487	0.2248	0.2489
2	0.2744	0.2365	0.2370	0.2369	0.2366	0.2191	0.2369
3	0.2719	0.2294	0.2310	0.2310	0.2310	0.2171	0.2301
4	0.2788	0.2272	0.2330	0.2329	0.2323	0.2165	0.2306
5	0.2794	0.2365	0.2357	0.2357	0.2322	0.2072	0.2319
6	0.2820	0.2274	0.2379	0.2352	0.2310	0.2081	0.2308
7	0.2830	0.2354	0.2409	0.2365	0.2338	0.2044	0.2364
8	0.2745	0.2308	0.2350	0.2300	0.2274	0.2020	0.2299
9	0.2825	0.2305	0.2326	0.2274	0.2245	**0.1915**	0.2274
10	0.2802	0.2325	0.2335	0.2279	0.2262	0.2087	0.2279

Table 5.5: Result from action research cycle 5, best value is marked in bold

A graph of the prediction can be seen in Figure 5.8, where the dotted line is the training data, the dashed line is the prediction made and the solid line is the actual value for this data. Here the prediction can foresee small trends in the data, but the error is still high.

5.6 Action Research Cycle 6

The sixth cycle was also performed at Ericsson AB, but the focus was pivoted from the inflow of defects to the backlog. A graph of the backlog can be seen in Figure 5.9. For confidentiality reasons the axis have been erased.

In Table 5.6, the result from the sixth cycle can be seen. The data was evaluated using 90% as training data and 10% as testing data. In this cycle the best model was the benchmark measure with a lag of 9. As in cycle 3, the inducer algorithms outperforms the instance based models. An explanation of this can be the dependence in the data. As in this KPI, the value of the previous time step is the same $\pm$ the delta in-/outflow. The inflow KPI additionally, the inflow of defects one time step is completely independent of the previous time step inflow.

Since no prediction could outperform the benchmark, no prediction graph is shown.

Lag	Bench-mark	Linear SVR	Linear Regression	Lasso	Ridge	KNN	Elastic Net
1	0.0120	0.0120	0.0134	0.0122	0.0130	0.0216	0.0122
2	0.0116	0.0116	0.0129	0.0121	0.0129	0.0250	0.0119
3	0.0111	0.0111	0.0129	0.0115	0.0126	0.0257	0.0114
4	0.0114	0.0114	0.0130	0.0117	0.0130	0.0267	0.0118
5	0.0117	0.0117	0.0133	0.0121	0.0132	0.0266	0.0121
6	0.0121	0.0121	0.0135	0.0124	0.0135	0.0257	0.0124
7	0.0125	0.0125	0.0143	0.0128	0.0141	0.0276	0.0128
8	0.0129	0.0129	0.0150	0.0132	0.0150	0.0262	0.0132
9	**0.0098**	*0.0100*	0.0120	0.0101	0.0120	0.0279	0.0101
10	0.0102	0.0102	0.0124	0.0103	0.0116	0.0267	0.0104

Table 5.6: Result from action research cycle 6

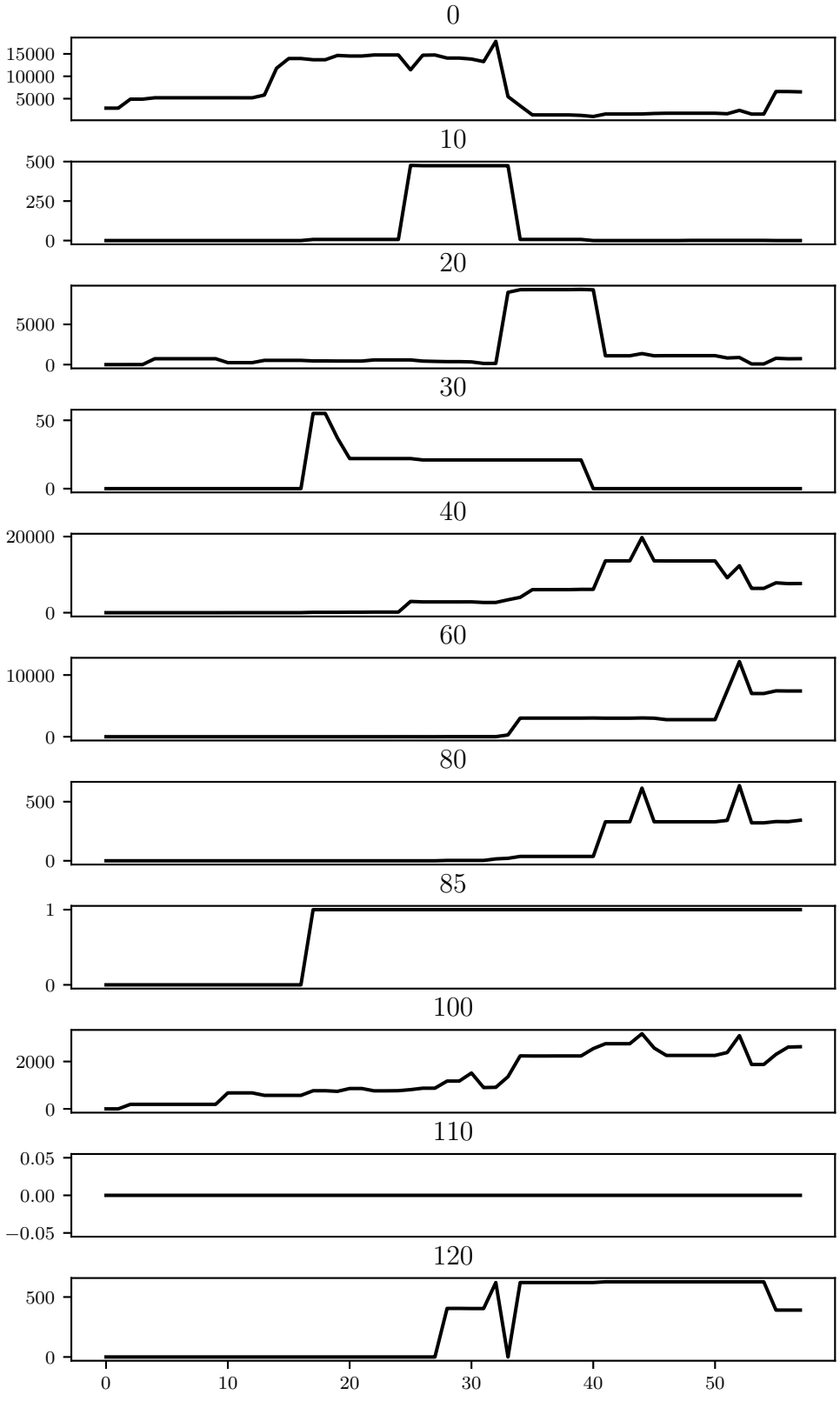

Figure 5.6: Status levels over time

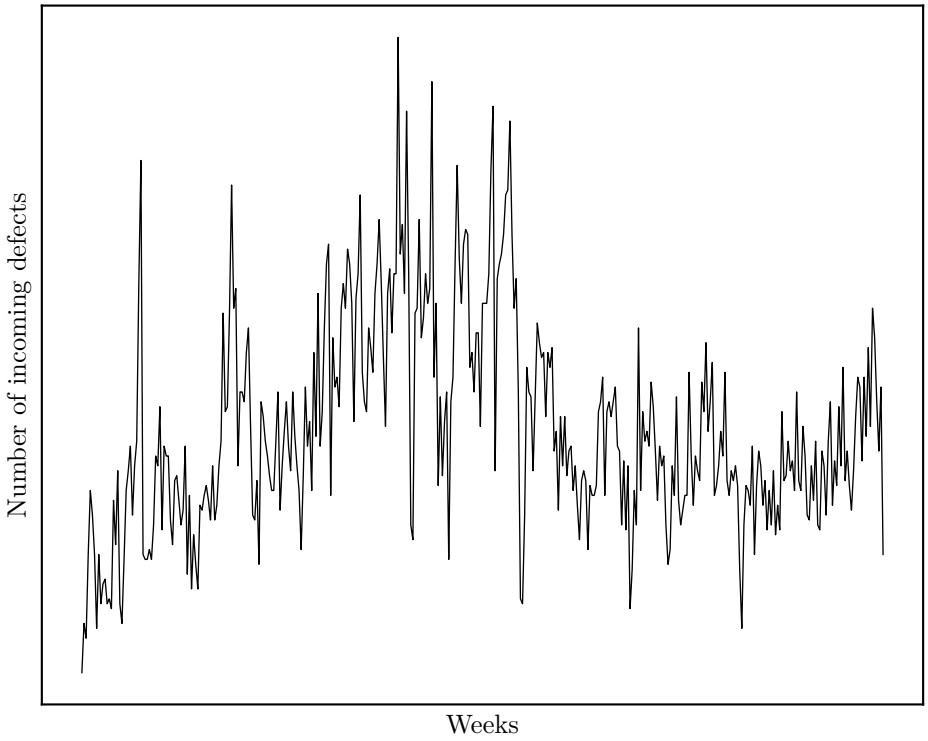

Figure 5.7: The defect inflow at Ericsson AB for a single product

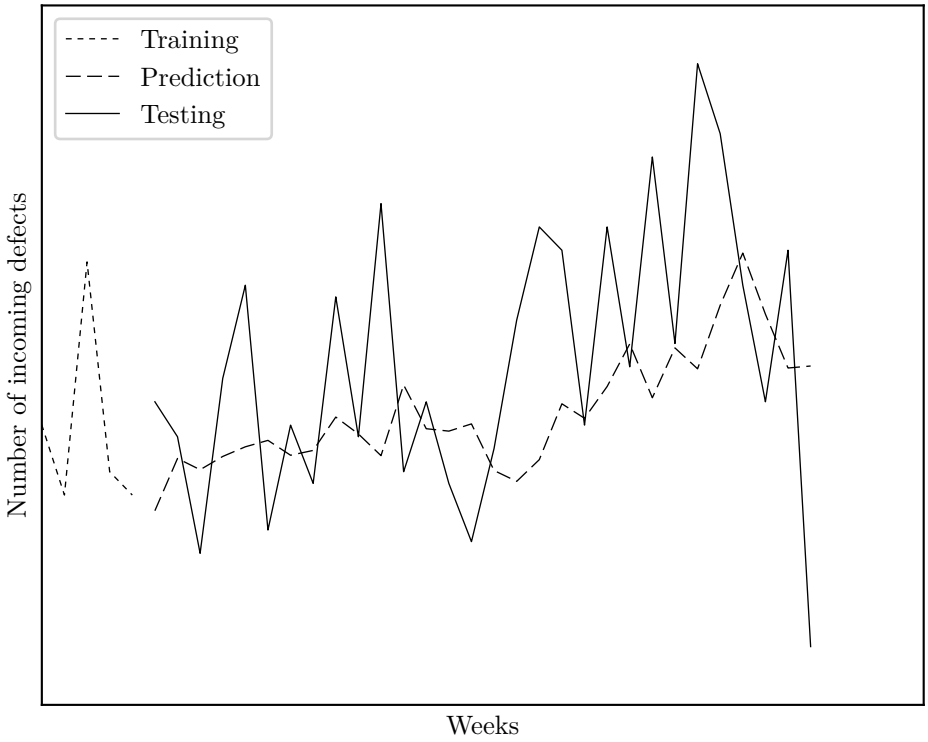

Figure 5.8: The defect inflow predictions for a single product

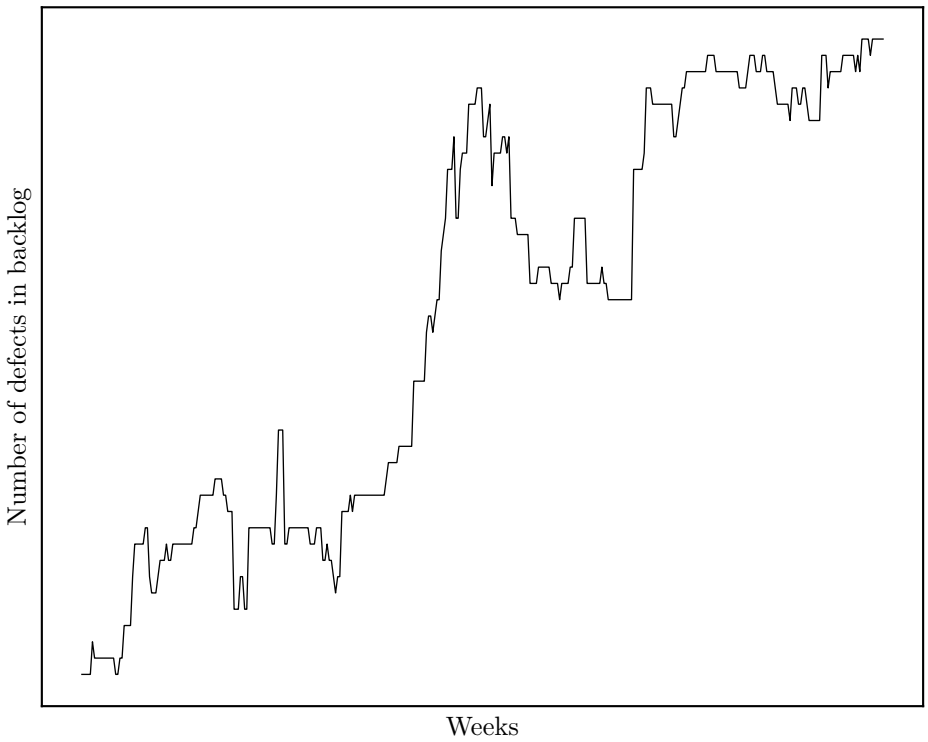

Figure 5.9: The defect backlog for a single product at Ericsson AB

6
Discussion

In this chapter, the results are discussed and compared to the literature.

6.1 Results

In this results section, the methods applied in the cycles is discussed. Research cycle 1 and 2 are the *Linear approach* and 3-6 are the *Rolling window approach*. Then the result of the cycles is discussed in pairs.

6.1.1 Linear approach

In the linear approach, a naïve approach was taken. This yielded only linear prediction on the data that proved to be poor estimates.

In this approach, the time dependency was not captured. As linear predictors does not utilize the dependency properly, the modeling of data was altered. This lead to the implementation of the rolling window model.

6.1.2 Rolling window approach

The rolling window model was used to force a dependency on time, as this opened up for applications of linear modeling of the data by relations between data points. Or to use a instance based learner to learn from previously seen patterns.

Since data is probably correlated, it can be tested by using a correlation matrix. A correlation matrix of the inflow can be seen in Table 6.1 and a correlation matrix of the backlog data can be seen in Table 6.2. The correlation matrix is made using the Pearson correlation function.

One reflection from the result from Table 6.1 and 6.2 is that the inflow is not as highly correlated as the backlog. This is logical, as the value for the inflow for one week is not related on the previous value. But the the value of the backlog is closely related to the to the previous value.

Table 6.3 shows the correlation matrix for the VCC KPI. Here, as in Table 6.2 (backlog), the correlation is higher than Table 6.1 (inflow). The reason behind the high correlation, is the value is dependent on the previous value as it is an average status. The rolling window from the fourth research cycle is not modeled as it would not add any additional information.

	W	W-1	W-2	W-3	W-4	W-5	W-6	W-7	W-8	W-9	W-10
W	1	0.68	0.59	0.49	0.51	0.47	0.42	0.39	0.38	0.38	0.39
W-1	0.68	1	0.69	0.6	0.49	0.52	0.48	0.43	0.4	0.38	0.39
W-2	0.59	0.69	1	0.69	0.6	0.5	0.52	0.48	0.43	0.41	0.39
W-3	0.49	0.6	0.69	1	0.69	0.6	0.5	0.52	0.49	0.44	0.41
W-4	0.51	0.49	0.6	0.69	1	0.69	0.6	0.5	0.53	0.49	0.44
W-5	0.47	0.52	0.5	0.6	0.69	1	0.69	0.6	0.5	0.53	0.49
W-6	0.42	0.48	0.52	0.5	0.6	0.69	1	0.69	0.6	0.5	0.53
W-7	0.39	0.43	0.48	0.52	0.5	0.6	0.69	1	0.7	0.61	0.5
W-8	0.38	0.4	0.43	0.49	0.53	0.5	0.6	0.7	1	0.7	0.61
W-9	0.38	0.38	0.41	0.44	0.49	0.53	0.5	0.61	0.7	1	0.7
W-10	0.39	0.39	0.39	0.41	0.44	0.49	0.53	0.5	0.61	0.7	1

Table 6.1: Correlation matrix for the Inflow with a lag of 10

	W	W-1	W-2	W-3	W-4	W-5	W-6	W-7	W-8	W-9	W-10
W	1	1	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.96	0.96
W-1	1	1	1	0.99	0.99	0.98	0.98	0.98	0.97	0.97	0.96
W-2	0.99	1	1	1	0.99	0.99	0.98	0.98	0.98	0.97	0.97
W-3	0.99	0.99	1	1	1	0.99	0.99	0.98	0.98	0.98	0.97
W-4	0.98	0.99	0.99	1	1	1	0.99	0.99	0.98	0.98	0.98
W-5	0.98	0.98	0.99	0.99	1	1	1	0.99	0.99	0.98	0.98
W-6	0.98	0.98	0.98	0.99	0.99	1	1	1	0.99	0.99	0.98
W-7	0.97	0.98	0.98	0.98	0.99	0.99	1	1	1	0.99	0.99
W-8	0.97	0.97	0.98	0.98	0.98	0.99	0.99	1	1	1	0.99
W-9	0.96	0.97	0.97	0.98	0.98	0.98	0.99	0.99	1	1	1
W-10	0.96	0.96	0.97	0.97	0.98	0.98	0.98	0.99	0.99	1	1

Table 6.2: Correlation matrix for the Backlog with a lag of 10

	W	W-1	W-2	W-3	W-4	W-5	W-6	W-7	W-8	W-9	W-10
avg	1	0.98	0.95	0.93	0.91	0.9	0.89	0.88	0.86	0.83	0.8
W-1	0.98	1	0.98	0.95	0.93	0.92	0.91	0.9	0.88	0.86	0.83
W-2	0.95	0.98	1	0.98	0.96	0.94	0.92	0.91	0.9	0.88	0.87
W-3	0.93	0.95	0.98	1	0.99	0.96	0.94	0.93	0.92	0.9	0.89
W-4	0.91	0.93	0.96	0.99	1	0.98	0.96	0.94	0.93	0.91	0.9
W-5	0.9	0.92	0.94	0.96	0.98	1	0.98	0.96	0.94	0.92	0.91
W-6	0.89	0.91	0.92	0.94	0.96	0.98	1	0.98	0.96	0.94	0.92
W-7	0.88	0.9	0.91	0.93	0.94	0.96	0.98	1	0.98	0.96	0.93
W-8	0.86	0.88	0.9	0.92	0.93	0.94	0.96	0.98	1	0.98	0.96
W-9	0.83	0.86	0.88	0.9	0.91	0.92	0.94	0.96	0.98	1	0.98
W-10	0.8	0.83	0.87	0.89	0.9	0.91	0.92	0.93	0.96	0.98	1

Table 6.3: Correlation matrix for the status KPI with a lag of 10

6.1.3 Research cycle 1 and 2

In the first cycle, there were no relative measure to compare against. This leaves the fitness of the prediction to the end user, as that person has to decide whether a MMRE of 0.12is acceptable at that scale. One leverage over the rolling window approach is the ability to predict longer periods of time. The downside is the linear generalization which yields a high error rate.

In the second cycle, the predictions can be compared to the first and they are nearly half as good as the first cycle, as the MMRE is 0.24. Here the objective was to predict the average status by predicting the number of parameters in each status bin. This would have been useful as this could provide a prediction on where a lot of parameters are.

Eliciting stakeholders at VCC and showing the prediction of the KPI a few comments were raised. The first regarded the suitability of predictions, stressing that long term predictions is useful for predicting the start of production for a project. The usefulness of these predictions was at the time low as new employees has not been accustomed to the workflow yet and the reporting of status changes could then be delayed. The more experienced employees had a more linear relationship in reporting status changes over time and could utilize the simple linear predictions.

6.1.4 Research cycle 3 and 4

The result from the third cycle, is the first cycle where rolling window was applied. In this cycle, predictions seemed surprisingly good in contrast to the previous cycle. Inspections of a plotted predictions showed that the prediction is a lagged true value, meaning that the prediction returned approximately the last known value. This lead to the introduction of the benchmark measurement. The benchmark measurement proved a good indicator on how low the error could be when just returning the previous value.

In the fourth cycle, the results were not as good as the third cycle. The difference is not massive, but it is noticeable. In this cycle a MMRE value over 1.0 is shown for the first time. Going back to the MMRE definition it shows that this is clearly possible and must mean that the delta value of the prediction and the true value is greater than the delta value of the true value and the x axis. Making the prediction be greater than 1.7 times the actual value.

An interesting observation is the outperforming of inducer based algorithms (Linear algorithms) to the instance based algorithms (KNN) in both of these cycles.

Comparing the accuracy of predictions from cycle 3 and 4 with cycle 1 and 2 shows a clear benefit of short term predictions. But the long term prediction of the first two cycles could be a useful measurement.

6.1.5 Research cycle 5 and 6

The last two cycles was performed at Ericsson AB inspecting their KPIs regarding defects. The first cycle inspected the inflow of defects. Comparing the MMRE of this prediction with others, an interesting observation is the relatively high error rate in both instance based algorithms and inducers. There is still a clear difference

through the data set showing that the best algorithm is the instance based. The lowest MMRE measured is 0.19and this is approximately four times larger than all other rolling window predictions. But the high error rate is still lower than the benchmark, making this prediction the most use full one.

The sixth, and last, cycle showed the lowest error observed in any of the previous research cycles. Here a MMRE below 0.01was achieved. One reason behind this low error could be the small changes in the data set, compared to the other data sets, as seen in Figure 5.9.

One observation is that the linear SVR matches the benchmark value in almost every prediction. Another observation is that the benchmarked error is not decreasing until the 9th lag, which is the lowest.

6.2 Related work

This section elaborates on the different approaches taken, as well as related problems found in literature. It also compares the current status of defect inflow predictions in literature, to the findings in this thesis.

6.2.1 Rolling window

The rolling window approach was taken in research cycle 3 to 6, as a way to model the time dependency. Similar approaches have been used in literature to study stock marked predictions with similarities to both the defect KPIs and the average status KPI [64].

The same work mentions the use of an index together with the stock price that is predicted [64]. This has a lot of similarities to the KPI used in cycle 1-4 (VCC), as previous projects were used together to predict a new one.

Previous attempts at defect inflow is found in the literature, where approaches similar to the rolling window has been applied. One approach uses data such as project planning and test statuses to predict the next week by using these values for previous weeks [65]. Another approach uses the defect inflow and the planning of the project and lag these variables for each week [66]. A third approach applies the rolling window to the inflow and outflow and uses the previous number of defects in the backlog to predict the backlog of a certain week [67]. A fourth approach applies the rolling window to the defect inflow of different stages in a project to predict the next time step [67].

6.2.2 Defects

Since the prediction of defects using machine learning is mostly focusing on classifying defect prone files or code segments, the same approach cannot be taken when predicting the number of defects at a given time [68, 13]. Even though there are some previous research that used to predict the total number of defects using Bayes Net, this involved many more features than just the backlog [69].

6.2.3 Defect backlog

There is also non-machine learning approaches to predicting the defect backlog [67], this approach uses Equation 6.1, where *db* is the defect backlog, *di* is the defect inflow, *do* is the defect outflow and *i* is the given time step.

$$db(i) = db(i-1) + \frac{di(i-1) + di(i-2) + di(i-3)}{3} - \frac{do(i-1) + do(i-2) + do(i-3)}{3} \tag{6.1}$$

Interpreting this equation in words: the next value, is the previously known value, plus the average inflow for the last three time steps, minus the average outflow for the last three time steps. This equation is reported to give a MMRE of 0.16, in contrast to the result in this thesis, using just the previous defect backlog reported a MMRE of lower then 0.01. Even though this measurement was performed on different data sets, the difference is remarkable.

6.2.4 Defect inflow

Predictions regarding the inflow can be derived from the software reliability growth model (SRGM). In SRGM, the cumulative number of defects is displayed in a project, the inflow is then the derivative of this model. The SRGM has been studied in different domains [70, 66] as well as long-term predictions [71]. SRGM can leverage previous project within the same organization to provide more accurate results [71].

One previous approach to defect inflow predictions is listed in [66]. In this paper, a MMRE of 0.52is achieved when predicting one week ahead. A study conducted a year later by the same authors introduced a method for predicting defect inflow based on e.g. test status and project planning [65]. Here a multivariate linear regression model was used, this yielded results around a MMRE of 0.24. Comparing the result of these two studies with the result in this thesis, a MMRE of 0.24was also achieved using only the previous inflow of defects and a multivariate linear regression model. In addition, this thesis also presents a MMRE of 0.19in the defect inflow prediction using the instance based learner KNN.

6.2.5 Status level KPI

The KPI used at VCC is constructed especially for that purpose, no equivalent measure was found in the literature.

6.3 Future work

To further the work with KPIs and machine learning predictions a few alternatives can be used. Below is some different approaches discovered during the development phase as well in discussions with people with machine learning experiences at Ericsson AB.

6.3.1 Further feature extraction

Using more features related to the defect inflow and defect backlog could provide even more accurate results. Even pre-processing the data differently could provide more accurate results. Using a normalization together with the defect inflow and adding previous projects to this could provide better defect predictions when applying KNN.

6.3.2 Extending the lag

The defect inflow shows longer trends then just 10 weeks back, therefore one approach can be to extend the lag variable that was set at a maximum 10 in this thesis to go further back. Going as high as 52, which would show trends over a year, could be a solution.

6.3.3 Convolutional Neural Networks

KPIs is a changing value over a time series, as identified. Applying the rolling window model opens up a possibility of using other approaches to predict future values. All KPIs can be modeled using Convolutional Neural Networks (CNN) [72]. The CNN selects a square of a matrix to train on, then using a weight function outputs a new value to the rest of the network. As the CNN can input 2d vectors and process them in 2d, it is very useful in image recognition. In the case of rolling window, selecting a square (or rectangle) representing a lagged time series could be used as input vector to the CNN. An illustration of this can be seen in Figure 6.1.

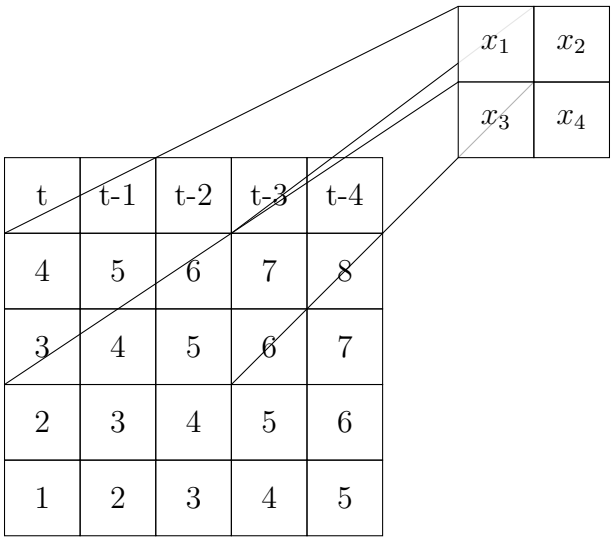

Figure 6.1: Illustration on how a CNN could be used in time series analysis

6.3.4 Recurrent Neural Networks

Another deep learning method, aside from CNN, that could be used is Recurrent Neural Networks (RNN) [73]. A recurrent neural network uses the structure of a

Neural Network, but adds a small memory of the previous inputs. This memory is a fed forward to the next cell, processing the next value in a series, which then decides what information to keep and what to discard. A common architecture for RNN is Long Short Term Memory (LSTM) [74], which makes it possible to model longer sequences of data in the RNN. This could be applied on the data without having to model this as a rolling window and instead see longer sequences of trends in the data.

6.3.5 Hidden Markov Model

Hidden Markov Models (HMM) [75] is used for modeling state transitions of an hidden entity by observations that can be derived from the hidden entity. Real world applications include Speech- and handwriting recognition.

In the case of the VCC and Ericsson AB KPI, state transitions could be useful information. The more applicable case is the VCC KPI, with variables changing status level. Some useful information could be to model the state transition matrix for different type of variables, as simpler variables would intuitively reach a higher status earlier than more complex variables. This would require more detailed data, such as the status change of each variable, but this was not easily accessible.

The Ericsson AB KPI could exploit the state transition by estimating how long a defect would remain in the defect database. This could yield information regarding the severity of the defect or the lead time of a fix of the defect.

6.4 Threats to validity

To test the validity of this thesis, four aspects are reviewed: the conclusion validity, the internal validity, the construct validity and the external validity [76].

6.4.1 Conclusion validity

The threat to conclusion validity in this thesis is discussed for only the defect inflow KPI as this is the only prediction that proved useful. The instance based learner outperformed the other prediction methods in each lag, so the conclusion that the instance based is superior when working with data with low dependency can be considered as valid in this case. The conclusion that this method is superior to previous methods is more doubtful as this thesis uses different data then the compared papers.

6.4.2 Internal validity

The internal validity of this thesis is discussed for each KPI. The internal validity for the status KPI used by VCC is the alteration made to the smaller project where the week number was doubled to be closer to the shape of the other projects. This was considered as a special case and suggested by an expert at VCC to apply this method. The fist KPI at Ericsson AB was the defect inflow. The internal validity threat is related to the data available, as a defect can be closed and opened again

without changing its ID, the actual defect inflow could be uncertain. This was mitigated by talking to experts at the company which explained it was on rare occasions and probably could be skipped. The second KPI at Ericsson AB was the defect backlog. Here, the internal validity is also related to the data set used in this thesis. As the backlog computation relied on the in- and outflow of defects, it has the same validity threat as previously discussed inflow. Adding to this is the threat of not reporting smaller defects. When simple and quickly fixed defects are discovered it can take longer time to report the defect then to fix it, they are usually fixed immediately.

6.4.3 Construct validity

Construct validity threat in this thesis can be found in the linkage between the different outcome of inducer algorithms versus instance based algorithms, and the correlation matrices. It is important to note that this is not a causation, but merely a correlation and a possible explanation.

Again, the comparison between the previous methods to perform inflow predictions and this method cannot be directly compared, but is more of an estimate that there is a successful way of predicting this.

6.4.4 External validity

The status KPI at VCC was unable to be tested further, as the projects to apply tests on was used in the prediction. The KPIs at Ericsson AB was only applied to one product and can therefore be subject to external validity threats and no generalizability can be claimed.

7

Conclusion

Applying machine learning to KPIs is a complex task as is shown. Depending on whether the data in the KPI has an underlying dependency or not, different algorithms have different suitability. The usefulness of accurate predictions in software engineering is important as this can both lower the cost and even out the effort for maintaining and developing software.

This thesis investigated one KPI form VCC regarding the status of calibration variables used in hardware components. Two approaches were taken to predict different aspects of the KPI. The first approach was to use linear regression on the data as is. The second approach was to use a rolling time frame to predict new values given a certain lag. The two aspects regarded the form of the KPI, the first aspect was the average status on a weekly basis, the second aspect was the number of variables at each status level.

The results from these predictions showed that the best approach to predict one week ahead was to use the previous value as prediction as this resulted in the minimal error. Using the linear approach, the results yielded a too high error to be used in development. As organization is moving towards a linear (average status wise) reporting in their development process, which could make this predictions useful in the future.

The second stage of this thesis was the KPIs investigated at Ericsson AB. The two KPIs was the defect inflow and the defect backlog. Here, the approach used at VCC was repeated using the rolling time frame. One interesting observation is the difference in error from both of these KPIs, the inflow has a much higher error rate when modeled with the rolling time frame then the backlog has. This was then investigated and a reason for this difference was the dependency within the data. A value in the inflow is not dependent on the previous value, as the number of defects reported one week does not have a relation the number of defects reported previous week. The opposite is true for the backlog, as the number of defects open in the backlog is for one week is highly correlated with the values of the previous week. This does not make inflow predictions useless, trends can be seen over longer periods of time and therefore a expansion of the lag could be a solution.

Predictions made showed a new approach for predicting the defect inflow using the instance based learner KNN, provided MMRE of 0.19. A better inflow prediction score could not be found in literature.

7.1 Research questions

This sections answers the research questions listed in the beginning of this thesis.

Q1) What data used to form KPIs is available and can be used as features and labels in a machine learning prediction?

The data used to form KPIs at VCC and Ericsson AB was the same data used to construct the KPIs. The data used was the week number and the status, defect inflow and defect backlog for that week. This was both applied without any transformation, and using a rolling window approach.

Q2) How to use machine learning algorithms to improve the current state of a KPI by predictions?

This process is explained in Chapter 4 by doing predictions on the KPI. The process includes data transformation to capture the time dependency using a rolling window, and applying a supervised regression algorithm. The best result was achieved using a rolling window and the instance based learner KNN on the defect inflow.

Q3) Given the available data sets at VCC and Ericsson AB – which KPIs can be improved? The prediction on the status KPIs at VCC did not improve the KPI as the predictions were inadequate.The defect inflow KPI at Ericsson AB was the KPI that proved better than the literature and was improved with predictions.

Bibliography

[1] M. Ochodek, M. Staron, D. Bargowski, W. Meding, and R. Hebig, "Using Machine Learning to Design a Flexible LOC Counter," *Workshop on Machine Learning Techniques for Software Quality Evaluation*, 2017.

[2] scikit-learn developers (BSD License), "Choosing the right estimator scikit-learn 0.18.1 documentation," 2016. [Online]. Available: http://scikit-learn.org/stable/tutorial/machine_learning_map/

[3] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, no. 3, pp. 249–268, 2007.

[4] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005.

[5] M. Staron, W. Meding, and K. Palm, "Release readiness indicator for mature agile and lean software development projects," *International Conference on Agile Software Development*, pp. 93–107, 2012.

[6] B. Marr, "Corporate performance measurement." *Controlling*, vol. 17, no. 11, pp. 645–652, 2005.

[7] M. Bourne, M. Franco, and J. Wilkes, "Corporate performance management," *Measuring Business Excellence*, vol. 7, no. 3, pp. 15–21, 2003. [Online]. Available: http://www.emeraldinsight.com/doi/10.1108/13683040310496462

[8] D. Christopher and F. David, "Innovations in performance measurement : Trends and research implications," *Journal of Management Accounting Research*, vol. 10, pp. 205–238, 1998.

[9] A. Abran, *Software Metrics and Software Metrology.* John Wiley & Sons, 2010.

[10] ——, *Software project estimation: the fundamentals for providing high quality information to decision makers.* John Wiley & Sons, 2015.

[11] A. Abran, A. Sellami, and W. Suryn, "Metrology, measurement and metrics in software engineering," *Proceedings - International Software Metrics Symposium*, vol. 2003-Janua, pp. 2–11, 2003.

[12] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 126–137, 1995.

[13] V. U. Challagulla, F. B. Bastani, I.-L. Yen, and R. A. Paul, "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques," *10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems*, pp. 263–270, 2005.

[14] M. Staron, W. Meding, K. Niesel, and A. Abran, "A Key Performance Indicator Quality Model and Its Industrial Evaluation," *Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pp. 170–179, 2016.

[15] M. Hobson, P. Graff, F. Feroz, and A. Lasenby, "Machine-learning in astronomy," *Statistical Challenges in 21st Century Cosmology*, vol. 306, no. 306, pp. 279–287, 2014.

[16] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, and V. Robles, "Machine learning in bioinformatics," *Briefings in Bioinformatics*, vol. 7, no. 1, pp. 86–112, 2006.

[17] S. Wang and R. M. Summers, "Machine Learning and Radiology," *Medical image analysis*, vol. 16, no. 5, pp. 933–951, 2012.

[18] F. J. Mulhern and R. J. Caprara, "A nearest neighbor model for forecasting market response," *International journal of forecasting*, vol. 10, no. 2, pp. 191–207, 1994.

[19] A. Weber and R. Thomas, "Key Performance Indicators - Measuring and Managing the Maintenance," *IAVARA Work Smart*, no. November, pp. 1–16, 2005.

[20] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting Empirical Methods for Software Engineering Research," *Guide to Advanced Empirical Software Engineering*, pp. 285–311, 2008.

[21] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5392560

[22] P. Louridas and C. Ebert, "Machine Learning," *IEEE Software*, vol. 33, no. 5, pp. 110–115, 2016.

[23] R. Kohavi and F. Provost, "Glossary of terms," *Machine Learning*, vol. 30, pp. 271–274, 1998. [Online]. Available: http://ukupa.org.uk/resources/glossary-of-terms/

[24] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[25] C.-c. Chang and C.-j. Lin, "LIBSVM : A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, pp. 1–39, 2011.

[26] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classiiers," *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, 1992.

[27] A. J. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[28] H. Chih-Wei, C. Chih-Chung, and L. Chih-Jen, "A Practical Guide to Support Vector Classification," pp. 1–16, 2003.

[29] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[30] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[31] G. H. GOLUB and C. VAN LOAN, "AN ANALYSIS OF THE TOTAL LEAST SQUARES PROBLEM," *SIAM Journal on Numerical Analysis*, vol. 17, no. 6, pp. 883–893, 1980.

[32] D. R. Cook, "Detection of Influential Observation in Linear Regression," *Technometrics*, vol. 19, no. 1, pp. 15–18, 1977.

[33] R. TIBSHIRAN, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[34] C. Saunders, A. Gammerman, and V. Vovk, "Ridge Regression Learning Algorithm in Dual Variables," *Proceedings of the 15th International Conference on Machine Learning*, vol. 98, pp. 515–521, 1998. [Online]. Available: http://eprints.soton.ac.uk/258942/

[35] H. Zou and T. Hastie, "Regularization and variable selection via the elastic-net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[36] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[37] E. Fix and J. Hodges, *Discriminatory analysis : nonparametric discrimination, consistency properties*. University of California, Berkley, 1951.

[38] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-Based Learning Algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] Scikit-learn, "4.3. Preprocessing data scikit-learn 0.18.1 documentation," 2016. [Online]. Available: http://scikit-learn.org/stable/modules/preprocessing.html#label-encoding

[41] T. Dietterich, "Machine learning for sequential data: A review," *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 1–15, 2002.

[42] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192–213, 2012.

[43] GitHub, "scikit-learn/sklearn/metrics/regression.py," 2017. [Online]. Available: https://github.com/scikit-learn/scikit-learn/blob/14031f6/sklearn/metrics/regression.py

[44] P. J. Rousseeuw and C. Croux, "Alternatives to the Median Absolute Deviation," *Journal of the American Statistical Association*, vol. 88, no. 424, pp. 1273–1283, 1993.

[45] GitHub, "Model evaluation," 2017. [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html

[46] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," *IEEE Transactions on Software Engineering*, vol. 29, no. 11, pp. 985–995, 2003.

[47] B. Kitchenham, L. M. Pickard, S. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure," *IEE Proceedings - Software*, vol. 148, no. 3, pp. 81–85, 2001.

[48] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: an analysis and review," *International Journal of Forecasting*, vol. 16, no. 4, pp. 437–450, 2000.

[49] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[50] A. P. Chan and A. P. Chan, "Key performance indicators for measuring construction success," *Benchmarking: An International Journal*, vol. 11, no. 2, pp. 203–221, 2004.

[51] M. J. Lebas, "Performance measurement and performance management," *International Journal of Production Economics*, vol. 41, no. 1-3, pp. 23–35, 1995.

[52] D. Paramenter, "Chapter 1," in *Key Performance Indicators: Developing, Implementing, and using Winning KPIs*. John Wiley & Sons, 2010, pp. 1–29.

[53] Oxford University Press, ""indicator, n." OED Online," 2017. [Online]. Available: http://www.oed.com/view/Entry/94420?redirectedFrom=indicator&

[54] S. Dransfield, N. I. Fisher, and N. J. Vogel, "Using Statistics and Statistical Thinking to Improve Organisational Performance," *International Statistical Review*, vol. 67, no. 2, pp. 99–122, 1999.

[55] E. M. Goldratt, *The Haystack Syndrome: Sifting Information Out of the Data Ocean*. North River Press, 1990. [Online]. Available: https://books.google.se/books?id=fvsOAQAAMAAJ

[56] R. S. Kaplan and D. P. Norton, "The Balanced Scorecard - Measures That Drive Performance," *Harvard Business Review*, vol. 70, no. 1, pp. 71–79, 1992.

[57] ——, "Putting the Balanced Scorecard To Work," *Harvard Business Review*, vol. 71, no. 5, pp. 134–142, 1993.

[58] M. Staron and W. Meding, "Ensuring Reliability of Information Provided by Measurement Systems," *International Workshop on Software Measurement*, pp. 1–16, 2009.

[59] K. Pandazo, A. Shollo, M. Staron, and W. Meding, "Presenting software metrics indicators: a case study," *Proceedings of the 20th International Conference on Software Product and Process Measurement (MENSURA)*, vol. 20, no. 1, 2010.

[60] I. X. e.-b. c. Collection) and I. X. S. (e book, *IEEE Std 15939-2008, IEEE Standard Adoption of ISO/IEC 15939:2007Systems and Software Engineering-Measurement Process*. IEEE, 2009, no. January.

[61] M. Staron and W. Meding, "Measurement-as-a-service - A new way of organizing measurement programs in large software development companies," *Lecture Notes in Business Information Processing*, vol. 230, pp. 144–159, 2015.

[62] V. Antinyan, M. Staron, A. Sandberg, and J. Hansson, "Validating Software Measures Using Action Research A Method and Industrial Experiences," *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 2016.

[63] C. Jones, "Measurement of Defect Potentials and Defect Removal Efficiency," *CrossTalk The Journal of Defense Software Engineering*, vol. 21, no. 6, pp. 11–13, 2008.

[64] Y. Wang and I. Choi, "Market Index and Stock Price Direction Prediction using Machine Learning Techniques: An empirical study on the KOSPI and HSI," *arXiv preprint arXiv:1309.7119*, pp. 1–13, 2013.

[65] M. Staron and W. Meding, "Predicting weekly defect inflow in large software projects based on project planning and test status," *Information and Software Technology*, vol. 50, no. 7, pp. 782–796, 2008.

[66] ——, "Predicting Short-Term Defect Inflow in Large Software Projects An Initial Evaluation," *EASE*, 2007.

[67] M. Staron, W. Meding, and B. Söderqvist, "A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation," *Information and Software Technology*, vol. 52, no. 10, pp. 1069–1079, 2010.

[68] T. Menzies, J. DiStefano, A. Orrego, and R. Chapman, "Assessing predictors of software defects," *Proc. Workshop Predictive Software Models.*, 2004.

[69] N. Fenton, M. Neil, W. Marsh, P. Hearty, D. Marquez, P. Krause, and R. Mishra, "Predicting software defects in varying development lifecycles using Bayesian nets," *Information and Software Technology*, vol. 49, no. 1, pp. 32–43, 2007.

[70] R. Rana, M. Staron, N. Mellegård, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Evaluation of standard reliability growth models in the context of automotive software systems," *International Conference on Product Focused Software Process Improvement*, 2013.

[71] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Torner, "Evaluating long-term predictive power of standard reliability growth models on automotive systems," *2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*, 2013.

[72] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997. [Online]. Available: http://ieeexplore.ieee.org/document/554195/

[73] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1993. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S089360800580125X

[74] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," *Interspeech 2014*, pp. 338–342, 2014.

[75] P. Blunsom, "Hidden Markov Models," *Lecture notes*, pp. 1–7, 2004.

[76] C. Wohlin, P. Runeson, M. Host, C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*. Springer US, 2000.