Master's Thesis

Prediction of refractory wear with Machine Learning methods

Manuel Forrer
manuel.forrer@student.tugraz.at

Institute for Theoretical Computer Science (IGI)
Graz University of Technology

Supervisor: Assoc.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Robert Legenstein
Assistant Supervisor: Dipl.-Ing. Gregor Lammer, RHI AG

March 12, 2012

“Who fights can lose, who doesn’t fight has already lost.”

Bertolt Brecht, [Peter Marischnig’s motto of life, †2005)]

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, ...	....
(date)	(signature)

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, ...	....
(Datum)	(Unterschrift)

Abstract

Steel manufacturing is a rather old business. Nevertheless, steel as a raw material is needed more than ever. Due to its importance, lots of projects investigate how to enhance the steel-making processes.

This thesis is part of a project called APO (Automatische Pflegeprogrammoptimierung) conducted by RHI AG, one of the world's leading refractory material manufacturer. The aim of APO is to optimize production processes in a steel-plant. Maintenance procedures should be automatized by developing an autonomous robot to relieve the workload of a steelworker in front of a very hot basic oxygen furnace (BOF) converter. Before developing a robot tremendous basic analyses have to be made. On the one hand, the refractory wear impacts on a BOF converter and its relationships to production process parameters have to be found. On the other hand, maintenance proposals have to be defined with respect to the amount, area and duration of a maintenance.

This thesis is concerned with the former part of APO that is, the automatic extraction of the relationships between production process parameters and refractory wear of a BOF converter. In order to achieve this, different machine learning approaches are investigated which analyse the data sets provided by the cooperating steel-plant HKM in Duisburg, Germany. Basically, this thesis tries to find relationships using linear methods as well as non-linear methods by applying supervised learning approaches. Particularly, regression analyses are performed starting with the simple *Linear Regression* in advance of investigating non-linear *Support Vector Regression* as well as applying *Gaussian Processes*.

Different areas of a BOF converter are analysed separately to determine whether or not suitable relationships between production process parameters and refractory wear can be observed. Moreover, this thesis evaluates the quality of the found relationships to detect the most suitable approach in terms of APO using statistical testing techniques. Finally, investigations are made to determine the influence of the most important production process parameters on the refractory wear. Performance analyses show the dependency of prediction quality for different converter areas and data set size. In general, *Gaussian Processes* perform slightly better than *Linear Regression*, while both outperform *Support Vector Regression* approaches.

Zusammenfassung

Die Stahlbranche ist eine sehr alte Branche. Trotzdem ist Stahl als Rohmaterial gefragter denn je. Aufgrund des großen Einflusses von Stahl investieren viele Firmen in Projekte zur Verbesserung der Stahlerzeugungsprozesse.

Diese Masterarbeit fungiert als Teil des Projektes APO (Automatische Pflegeprogrammoptimierung) das von der Firma RHI AG ins Leben gerufen wurde. Die RHI AG ist einer der führenden Feuerfestmaterialhersteller der Welt. Das Ziel von APO ist es Pflegeroutinen zu optimieren und zu automatisieren. Aus diesem Grund soll ein Roboter entwickelt werden, der den optimalen Pflegezeitpunkt vorschlägt und die Pflege bei Bedarf auch selbst direkt durchführt. Dadurch sollen die gesundheitlichen Risiken der Stahlarbeiter, die bei der Pflege direkt vor dem über 1000° Celsius heißen Stahlkessel auftreten, minimiert werden. Um so einen Roboter entwickeln zu können, ist es zuvor notwendig, die Zusammenhänge des Stahlerzeugungsprozesses zu erlernen um Pflegeprogramme automatisiert erstellen zu können.

Das Erkennen der Zusammenhänge zwischen Produktionsparametern und dem Feuerfestverschleiß ist die Hauptaufgabe dieser Masterarbeit. Verschiedene Ansätze aus dem Bereich des maschinellen Lernens werden betrachtet. Als Basis dienen die Datensätze die von dem Stahlwerk HKM aus Duisburg, Deutschland zur Verfügung gestellt werden. Basierend auf dem Prinzip des überwachten Lernens werden lineare als auch nicht lineare Methoden betrachtet. Diese Masterarbeit versucht Zusammenhänge durch die gezielte Anwendung von *Linearer Regression*, *Support Vector Regression* oder *Gaußschen Prozessen* zu erlernen.

Im Detail werden verschiedene Bereiche eines Stahlkonverters separat analysiert, um etwaige Zusammenhänge zwischen Produktionsparametern und Feuerfestverschleiß herauszufinden. Die Qualität der gefunden Zusammenhänge wird durch statistische Tests evaluiert, um die am besten funktionierende Methode zu bestimmen. Schlussendlich werden die Einflüsse der einzelnen Produktionsparameter auf den Feuerfestverschleiß noch gesondert betrachtet und extrahiert. Die Ergebnisse der Analysen zeigen, dass die Vorhersagegenauigkeit sehr stark von der betrachteten Region im Konverter sowie der zur Verfügung stehen Datenmenge abhängt. Im Allgemeinen liefern *Gaußsche Prozesse* das beste Ergebnis. Die *lineare Regression* funktioniert zumeist nahezu ähnlich gut. Die *Support Vector Regression* Ansätze liefern das schlechteste Ergebnis.

Acknowledgements

This thesis was conducted at the Institute for Theoretical Computer Science in association with RHI AG and HKM. I would like to thank the RHI AG, especially Gregor Lammer for giving me the opportunity of this research. Apart of Gregor, special mention goes to further RHI AG colleagues Christoph Jandl, Ronald Lanzenberger and Alexander Cepak who contributed to this thesis. Additionally, i wish to show my gratitude to the steel-plant HKM for providing their data set otherwise this thesis would not have been possible.

I especially wish to thank my supervisor, Prof. Robert Legenstein. I am indebted to his guidance in almost insolvable situations as well as his invaluable help and feedback throughout this thesis.

Last but not least, i would like to thank my parents Ingrid and Jakob, my siblings as well as my cousin Sebastian and especially my fiancé Elisabeth for the extensive support during my entire studies.

Manuel Forrer

Graz, Austria, March 2012

Table of Contents

List of Figures

List of Tables

List of Listings

Acronyms and Abbreviations

(FeO)	ferric oxide content in slag
(MgO)	addition of magnesium oxide carriers
[Mn]	manganese content of hot metal
[Si]	silicon content of hot metal
ANN	artificial neural network
AOC	area over the curve
APO	Automatische Pflegeprogrammoptimierung $\rightarrow$ intelligent maintenance optimization
ARIMA	autoregressive integrated moving average models
AUC	area under the curve
availability	heats per day
basicity	proportion between calcium oxide (CaO) in slag and silicon dioxide (SiO_2) in slag
BOF	basic oxygen furnace
BPNN	back-propagation neural networks
CG	conjugate gradient
CumScore	cumulative score
CV	cross validation
Fe	ferric oxide content in slag
FeSi	addition of ferro-silicon
GA-SVR	genetic algorithm support vector regression

Gefäß Nr.	Gefäß Nummer $\rightarrow$ vessel number
GP	Gaussian process
GPLS	generalized partial least squares
GPLS-GP	generalized partial least squares Gaussian process
GPML	Gaussian processes for machine learning
HKM	Hüttenwerke Krupp Mannesmann
ICA	independent component analysis
KKT	Karush-Kuhn-Tucker
KV	Konverter $\rightarrow$ converter
KV-Alter	Konverteralter $\rightarrow$ heat number
LaCam	laser camera
LD	Linz-Donawitz
liegezeit	time between end of blowing and beginning of tapping
LR	linear regression
MAE	mean absolute error
MLP	multi-layer perceptron
Mn_RE	manganese content of hot metal
MSE	mean squared error
MTWGP	multi-task warped Gaussian processes
PCA	principal component analysis
PLS	partial least squares
PSO	particle swarm optimization
RBF	radial basis function
REC	regression error characteristic
RGA	real-value genetic algorithm
RMSE	root mean squared error

ROC	receiver operating characteristic
Si_RE	manganese content of hot metal
SNR	Schmelznummer $\rightarrow$ unique heat id
SVM	support vector machine
SVR	support vector regression
temp_abstich	temperature at the end of blowing
WGP	warped Gaussian process

Teil I.

INTRODUCTION

1 Introduction

Contents

This chapter explains the basic facts of a steel-making process, focusses on the motivation of this thesis, defines some terminology used in steel-making and introduces the involved companies briefly. Chapter 2 deals with the theoretical background of the applied approaches and focusses on past researches using similar approaches. The analysed data set and necessary data pre-processing steps are introduced in Chapter 3. Moreover, the approaches applied in this thesis are covered, particularly. Chapter 4 focusses on the implementation of data pre-processing and approaches. The analyses results and performance evaluations are discussed in Chapter 5. Finally, Chapter 6 concludes this thesis and provides outlook for future work.

1.1. Steel-making process

From hot metal to steel. In the steel-making process liquid hot metal delivered from blast furnace and metal scrap is charged into a huge vessel called converter. Then the Linz-Donawitz (LD) process also called basic oxygen furnace process (BOF) is applied. Using lances, oxygen is blown onto the liquid melt arising some chemical reactions. Correctly applied oxygen blowing extracts several disturbing elements from the hot metal/scrap melt. A complete pass starting from filling the converter with hot metal and scrap until the pouring out of clean steel lasts approximately 45 minutes. Specialists call such a pass heat. Once a heat is completed, the liquid crude steel is tapped into a ladle where it is de-oxidised and pre-alloyed ahead of transporting it to secondary metallurgy using a steel ladle. The secondary metallurgy processes the steel to its desired chemical analysis. Figure 1.1 sketches a general overview of a typical steel-making process from

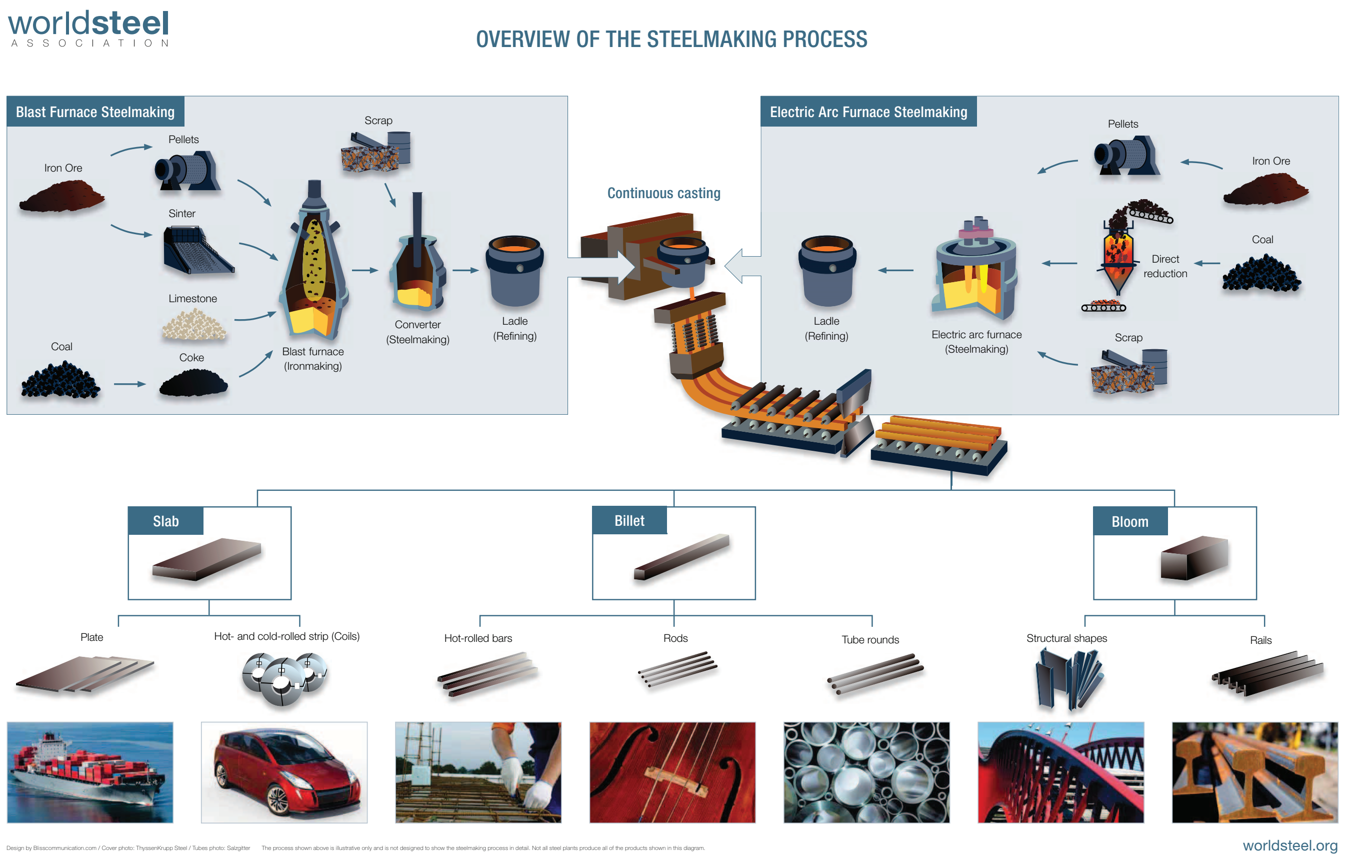

worldsteel ASSOCIATION
OVERVIEW OF THE STEELMAKING PROCESS
Blast Furnace Steelmaking
Iron Ore
Pellets
Sinter
Limestone
Coal
Coke
Scrap
Blast furnace (Ironmaking)
Converter (Steelmaking)
Ladle (Refining)
Continuous casting
Electric Arc Furnace Steelmaking
Pellets
Iron Ore
Coal
Direct reduction
Scrap
Electric arc furnace (Steelmaking)
Ladle (Refining)
Slab
Billet
Bloom
Plate
Hot- and cold-rolled strip (Coils)
Hot-rolled bars
Rods
Tube rounds
Structural shapes
Rails
Design by Blisscommunication.com / Cover photo: ThyssenKrupp Steel / Tubes photo: Salzgitter
The process shown above is illustrative only and is not designed to show the steelmaking process in detail. Not all steel plants produce all of the products shown in this diagram.
worldsteel.org

Figure 1.1.: Overview of the steel-making process. (Source: [World Steel Association, 2012])

the raw material to different high-quality end products mentioning blast furnace steel-making as well as electric arc furnace steel-making. Moreover, applications of different steel end products are shown.

Temperatures challenge converters. Different setups during a heat cause different temperature levels. Temperatures can reach peak values up to 1800° Celsius during a steel-making process. Such high temperatures challenge a steel converter. For this reason, vessels are lined with refractory material. Figure 1.3(a) shows a lined steel converter. However, refractory wears out as well as time goes by. In fact, refractory wear differs in speed and amount from area to area of a converter. Thus, almost every steel-plant invests time and money in converter maintenance.

Maintenance improves costs of steel making process. Maintenance draws steelworks attention in terms of optimizing their steel-making process. Today steelworkers can apply a various range of repair methods that mainly differ in terms of speed and applicability. Among others, gunning repair is one of the most important maintenance methods. Maintenance approaches will be covered in Section 1.2 in detail.

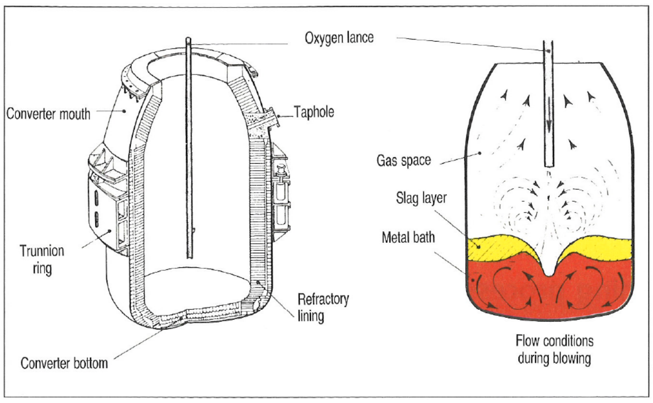


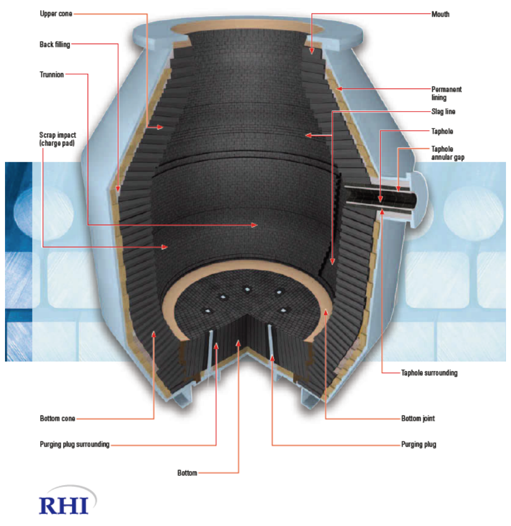
Figure 1.2.: Schematic visualization of BOF oxygen blowing process. The oxygen lance is launched on top of the hot and liquid slag/metal mixture ahead of starting the BOF blowing process. (Source: [Bolbrinker and Verein Deutscher Eisenhüttenleute, 1992])

Summing up, a huge amount of steps have to be passed through starting from the raw material arch to finally end up with the high-quality steel product.

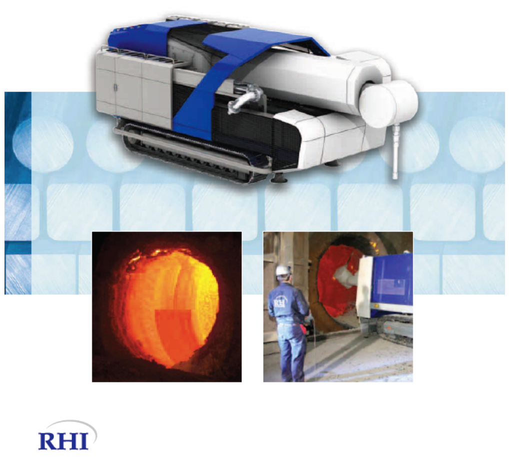
1.2. Terminology

The steel-making principle was introduced in Section 1.1. This section explains the most important terminologies covered in this thesis.

Converter. One of the main parts of a steel-making process is the converter. In general, converters are vessels in which the basic oxygen furnace (BOF) process is applied. Figure 1.2 shows a schematic visualisation of the BOF blowing process. Converters differ in shape and size from steel-plant to steel-plant. Their height varies from 5 up to more than 11 meters. Moreover, the converter volume differs as well. Capacities from 60 to 400 tons/heat exist. During a BOF process very high temperatures arise. The steel shell and the permanent lining is too weak to survive such conditions. Thus, a converter is additionally bricked with refractory material. These bricks are products varying in shape, size and quality. Different bricks are used for different areas inside a converter. This specific brick arrangement, called balanced lining, is planned in advance by refractory specialists in a so-called BOF lining concept. It is a continuous process in order to achieve the optimal lining in terms of costs and performance. Moreover, different bricks and different stresses through a heat results in different refractory behaviour. For this reason, areas showing similar refractory wear characteristics are grouped. Figure 1.3(a) shows a typical converter and indicates characteristic refractory wear areas as well [RHI AG, 2011a].



(a) Converter lined with refractory material. Characteristic converter areas indicated.



(b) Steelworker performing gunning maintenance by manually controlling a robot.

Figure 1.3.: Exemplary converter shape and maintenance procedure (Source: [RHI AG, 2011a])

Campaigns consist of many heats. Once a new converter lining is commissioned the initial heat is performed. Hot metal and scrap are charged and a lance is used to blow oxygen onto the melt inferring chemical reactions. The affinity of the melt to oxygen causes extraction of disturbing elements. The crude steel finally is poured out through a taphole into a steel ladle. This process is called tapping and is done by tilting the converter. During a tapping process the crude steel is pre-alloyed. The procedure of tilting the converter and pouring out the crude steel into a steel ladle lasts from 8 to 4 minutes depending on the taphole age. This time is called run-out-time. The steel ladle is used to transport the crude steel to the secondary metallurgy, processing it to its desired final chemical analysis. The whole procedure from one tapping to the next is called a heat and lasts approximately 45 minutes. Heats are joined together to define a campaign. Depending on the steelworks strategy campaigns comprises 400 to 5000 heats in Europe and more than 10000 heats in America or Asia. By definition, a campaign starts with a new converter lining and ends just ahead of the next lining [Jandl, 2011].

Maintenance objectives. Looking at Figure 1.4 one can see that thermal, chemical as well as mechanical stresses influence the refractory wear of a converter. Since the different challenges to a converter and its refractory material are known, maintenance automatically draws the attention. Converter maintenance objectives faced of steel producers are

- reduced specific brick wear
- increasing breakout safety
- repair of areas subject to premature wear
- longer and predictable lining lifetime
- observance of scheduled lining cycles.

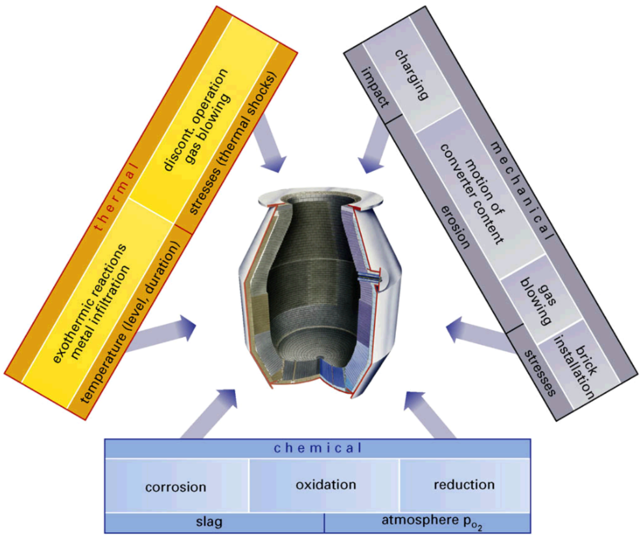


Figure 1.4.: BOF wear influences adapted from [Jandl, 2011]

In general, lining maintenance varies from almost zero to quite intensive repair after each heat from steel-plant to steel-plant. Several maintenance approaches are available.

No matter which approach is applied - the price/performance ratio and usability are the prime factors of an maintenance approach application [RHI AG, 2011b].

Various maintenance approaches exist. Generally speaking, BOF maintenance can be separated in two main categories

(1) maintenance using refractory products

(2) slag maintenance.

Standard gunning approaches using hand lance, the use of self-flowing mixes or manipulator gunning using a shooter belong to (1) and approaches like splashing, washing or forming can be assigned to (2). In fact, each steel-plant is interested to minimize the human influence during refractory maintenance and the reduction of physical stress on operators [RHI AG, 2011b].

In the following few typical BOF maintenance methods will be explained. One of the most used maintenance methods is manual or manipulator gunning. It uses special basic gunning mixes which are targeted on preworn converter areas to extend vessel lining. Nowadays, the parameters regarding time, amount and area of the gunning maintenance are mainly based on experience and laser scans. Concerning the experience-based maintenance, on the one hand, necessary repair processes can be missed and on the other hand, time/money is wasted due to senseless maintenance. Figure 1.3(b) shows a manipulator gunning application. As a matter of fact, maintenance reduces production time. Increasing the maintenance speed is one solution. Automatic gunning has the advantage of increased mix throughputs as well as improved gunning precision [RHI AG, 2011b]. A fully automatic maintenance robot performing automatic gunning is a future aim of the APO project, introduced later in this section.

Another approach is slag splashing developed 1982 in the United States. Once a converter is tapped, the remaining slag in the vessel is used for maintenance. The slag is splashed using high-pressure nitrogen jet to different converter areas within 2 to 5 minutes. The remaining excess slag after splashing is poured out before charging [RHI AG, 2011b].

Various other maintenance methods exist. [RHI AG, 2011b] explains many of them in detail.

Different converter areas require different maintenance methods. Experts in [RHI AG, 2011b] evaluated different maintenance methods for common refractory wear areas of a converter. The analysis results are shown in Table 1.1. Appropriate methods are indicated with (+), good methods with (++) and ideal methods with (+++). Blank cells indicate areas which cannot be maintained by the specific method.

Summing up, analysis results show that no method is ideal for all requirements. Thus, each steel-plant uses a combination of different maintenance approaches.

	Method	Machinery investment cost	Physical stress on operator	Lipring	Upper Cone	Charge Pad	Bottom joint	Bottom with plugs	Bottom without plugs	Tap pad	Taphole re-placement	Trunnion	Lower Cone
Slag maintenance	Modified (MgO) satu-rated slag	None	None		+	+	+	+	+	++		+	+
	Slag splashing	High	None			+	++	+	+++	++		++	++
	Slag washing	None	None			++	++	+	++	++			+
	Slag foaming	None	None			+	++	+	++	++		+	+
Maintenance using refractory products	Hand gunning	Low	High	+++	+	+	+	+	+	+	+++	+	+
	Manipulator gunning	Medium	Medium	++	++	++	++	++	+	++	++	++	++
	Automated gunning	High	Low	++	+++	+++	+++	++	++	+++	++	+++	+++
	Hot repair mix	None	Low		+	+++	++	+	+++	++			+
	Patching with bricks	None	Low		+	+++	++		+	+			+

Table 1.1.: BOF maintenance matrix. (Source: [RHI AG, 2011b])

1.3. Motivation

As everywhere but especially in the steel-working business it holds: time is money. Thus, RHI AG is interested in optimizing the maintenance program by developing a new maintenance robot. In doing so, it is required to know the exact refractory wear of each converter area in advance. Based on these predictions an optimized maintenance schedule can be proposed and applied by the maintenance robot fully automatically. RHI AG calls this project 'Automatische Pflegeprogrammoptimierung' (APO) which stands for *intelligent maintenance optimization*.

Even though, refractory researches are aware of the steel industry trends

- Maximum production
- Improving steel quality (e.g. clean steel)
- Minimization of specific refractory costs per ton steel [Jandl, 2011].

Aim of this thesis. One of the key features of APO is the refractory wear knowledge in advance. This thesis investigates different machine learning methods to achieve refractory wear knowledge. The main goal is to find relationships between the refractory wear and production process parameters. Starting with easy *Linear Regression (LR)* analyses, even more complex non-linear regression analyses using *Support Vector Regression (SVR)* or *Gaussian Processes (GP)* is applied and evaluated.

Since refractory wear differs for different converter areas and different plants, the converter will be analysed separately for each area subjected to similar wear characteristics. Moreover, more than 100 production process parameters control the steel-making process. For simplicity, the subset that has the most influence and can easily be changed will be chosen to perform regression analyses. Further on, the different areas will be analysed in detail using several approaches to construct a model of the process. The goal is to achieve a sense-full model resulting in proper predictions of future converter behaviour. Particularly, the refractory lining wear or in other words the converter evolution should be predicted.

Having a model providing the described predictions leads to various benefits like

- designing best fitting lining and maintenance concepts for every individual vessel
- real time system to assure optimum lining maintenance and targeted lining lifetime
- using scientific conclusions as a basis for refractory development.

Basically, all mentioned analyses are done on a data set comprising laser measuring data of the remaining converter lining thickness, on the one hand, and production process data, on the other hand. This data set will be provided from the cooperating steel-plant HKM introduced in Section 1.4. Section 3.1 introduces the specific APO approaches more detailed.

1.4. Involved companies

RHI AG. RHI AG is world leader in refractory technology. Refractory materials are indispensable for industrial processes exceeding 1200° Celsius to protect production units against thermal, mechanical and chemical stress. RHI AG is a global provider of high-quality refractory products including shaped products (refractory bricks) as well as unshaped products (refractory mixes). Approximately 50% of their products are based on captive materials like magnesite and dolomite. RHI AG has over 8000 employees at 33 production sites and more than 70 sales offices in four continents. Moreover, RHI AG produces approximately 2 million tons of refractory bricks and mixes as well as other refractory materials per year. Revenues of € 1,280.8 million have been recorded in the first three quarters of 2011. Their headquarters is located in Vienna [RHI AG, 2012]. In this thesis RHI AG acts as purchaser and is the founder of the whole project APO.

HKM. HKM is an abbreviation of Hüttenwerke Krupp Mannesmann. It is a steel-plant located in Duisburg in the Ruhr region, known as steel-making heart of Germany. HKM has been founded in January 1990 by Krupp Stahl AG and Mannesmannröhren-Werke AG [Hüttenwerke Krupp Mannesmann, 2002]. HKM has more than 3000 employees and produces approximately 5.6 million tons crude steel shipping capacity per year[1]. From an infrastructural point of view, HKM has two blowing stands and uses three change vessels. Thus, two converter vessels are used for production purposes whilst one vessel is equipped with a new refractory lining. Each converter has a capacity of 280 tons. In this thesis HKM acts as steel-plant supporting the thesis with their production process data and laser measurements.

Ferrotron. Ferrotron®, a division of MINTEQ International GmbH has been successfully developing laser technology since 1985. Nowadays, Ferrotron® is one of the market leaders. Among other systems, Ferrotron® develops a measurement system for non-contact measurement of refractory linings in metallurgical vessels. Principally, a laser scanner performing rapid scanning of the vessel using a pulsed laser beam which is deflected by a rotating mirror system. As a result, a three dimensional model for the object's inner surface is achieved. In other words, it is a kind of **La**ser **Cam**era giving the measurement system the name LaCam®. Ferrotron® produces fixed as well as mobile LaCam® systems [MINTEQ International GmbH - FERROTRON Division, 2009]. In this thesis Ferrotron® provides products to perform laser measurements at the steel-plant HKM based on which analyses are investigated. HKM is equipped with fixed LaCam® measurement units.

[1] http://www.hkm.de/english/the-enterprise/hkm-overview.php (Date: 20.01.2012)

Teil II.

THEORETICAL BACKGROUND

2 Related Work

Contents

This chapter presents former researches using the same methods we are about to use. Moreover, researches from similar as well as completely different domains are investigated. In order to understand the used concepts the theoretical background will be covered firstly.

2.1. Theoretical background

Before looking at former applications of

- *Linear Regression (LR)*
- *Support Vector Regression (SVR)*
- *Gaussian Processes (GP)*

the theoretic principles will be explained briefly. Sections 2.1.3, 2.1.4 and 2.1.5 introduce the most important facts about these methods. Firstly, some general characteristics relevant to all applied methods will be explained.

2.1.1. Supervised learning

In general, all methods covered in this thesis are supervised learning methods. In contrary to unsupervised learning methods, a supervised learning approach consists of training examples where each example is a pair of input and desired output/target.

Properties and characteristics. A very general characteristic of learning based pattern recognition is used. Thereby, the learning approach is split in two phases, namely

(1) training phase

(2) testing phase.

The main goal of the training phase is to find the optimal model-parameters. In other words to find the optimal weight vector $\boldsymbol{w}$. A simple approach to the problem of determining the model-parameters is to minimize the *sum-squared error function* $E_{\boldsymbol{w}}(L)$ shown in Equation 2.2 [Bishop, 2007]. The training set L comprises a set of input vectors $\boldsymbol{x}_n$ where $n = 1, \ldots, N$ together with a corresponding set of target vectors $\boldsymbol{t}_n$

$$L = \langle \langle \boldsymbol{x}_1, \boldsymbol{t}_1 \rangle, \langle \boldsymbol{x}_2, \boldsymbol{t}_2 \rangle, \ldots, \langle \boldsymbol{x}_N, \boldsymbol{t}_N \rangle \rangle. \tag{2.1}$$

Given this training set L, one approach is to minimize an error function such as the *sum-squared error function*

$$E_{\boldsymbol{w}}(L) = \frac{1}{2} \sum_{n=1}^{N} \left\| \boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w}) - \boldsymbol{t}_n \right\|^2 . \tag{2.2}$$

In other words, for each input, the output of the used approach should be as close as possible to the corresponding target. Finally, the weight vector $\boldsymbol{w}$ is found by solving the optimization problem.

Once the optimal model-parameters (weights) to represent the function of interest are found one moves on to the testing phase. This is the actual domain of interest or application part. Thus, the trained model is applied to a set of test data and results in a hopefully proper solution of the task.

***Underfitting* and *Overfitting*.** Generally, statistic learning methods can suffer from either *underfitting* or *overfitting*. On the one hand, a model which is not sufficiently complex to fully represent the target function leads to *underfitting*. On the other hand, a too complex model may for instance also fit to noise, not just to the signal of interest, leads to *overfitting*. The *overfitting* problem is also well known as generalization problem. This effect is illustrated in Figure 2.1 where a too complex model is used in a regression task to fit a line. Since *overfitting* can easily lead to bad predictions it should be avoided. The best way to avoid *overfitting* is to use lots of training data.

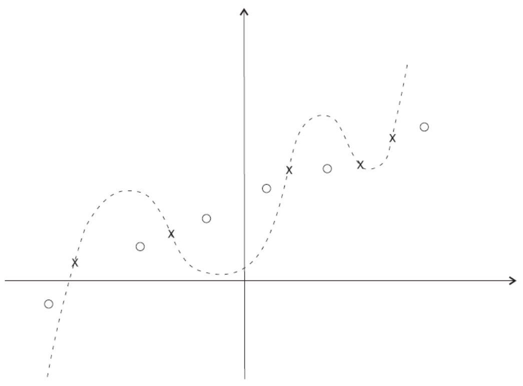


Figure 2.1.: Learning methods can suffer from either *under-* or *overfitting*. Not sufficiently complex models lead to *underfitting*. This plot shows the effect of *overfitting* where a too complex model is applied [Maass, 2008].

2.1.2. Classification vs. Regression

Recently introduced supervised learning approaches can be used to solve classification tasks as well as for regression problems.

The aim of a classification task is to separate objects into classes or categories. Objects having the same properties or characteristics will be collected to one class. In other words classification is the process of finding the correct classes for new feature vectors.

On the other hand in a regression task, one attempts to determine the relationship between dependent variables and a series of other changing variables. Regression is the supervised learning approach applied in this thesis.

2.1.3. Linear Regression

Linear Regression (LR) is one of the simplest approaches of regression and a widely used method. Some background information and the theoretical principle of *LR* will be introduced in the following.

Theoretical principle. The prediction of the value of one or more continuous target variables based on the given value of a D-dimensional vector $\boldsymbol{x}$ of input variables is the main goal of regression as stated in [Bishop, 2007].

One of the simplest approaches for regression is *LR*. *LR* involves a linear combination of the input variables and leads to a straight line y as an output like

$$y = w_0 + w_1 x \tag{2.3}$$

where w_0 is the intercept and w_1 the slope. Assuming the difference between the target

y and the straight line is denoted as an error ϵ, a more plausible model is

$$y = w_0 + w_1 x + \epsilon. \tag{2.4}$$

Equation 2.4 is called a *LR* model [Montgomery and Peck, 1992]. A *Linear Regression* model like

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D \quad \text{where} \quad \mathbf{x} = (x_1, \ldots, x_D)^T \tag{2.5}$$

with more than one model-parameter is called a *Multiple Linear Regression (MLR)* model estimating a plane-shaped target.

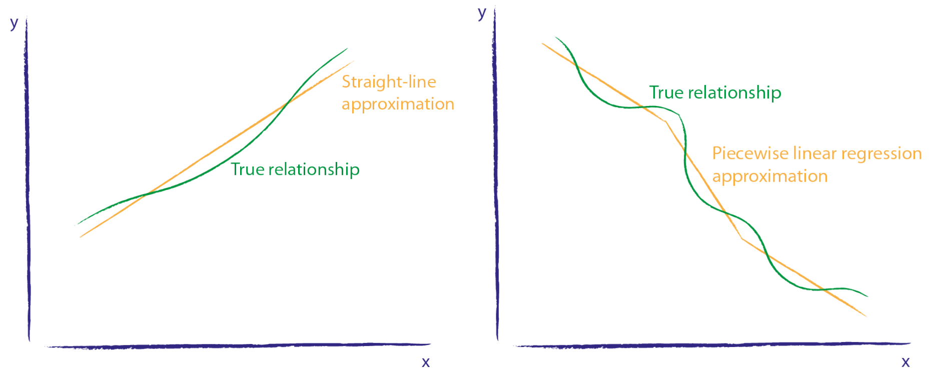
As for the *LR* the model-parameter vector $\boldsymbol{w}$ comprises the intercept w_0 and the slopes $w_1, \ldots, w_D$ are usually called regression coefficients. These model-parameters are assumed to have zero mean and an unknown variance σ^2. Additionally it is assumed that the regression coefficients are uncorrelated [Montgomery and Peck, 1992].

This model has one important key property, namely, it is a linear function of the model-parameters $w_0, \ldots, w_D$. As stated in [Bishop, 2007] the fact that it is a linear function of the input variables $\boldsymbol{x}$ as well imposes significant limitations to the model. For this reason, the model will be extended by considering linear combinations of fixed non-linear functions of input variables later in Section 2.1.4.

Model-parameter estimation. As in all supervised learning tasks, an important objective is to estimate the unknown model-parameters. In literature, this process is often called fitting the model to the data [Montgomery and Peck, 1992]. In *LR* a very common method is least-squares estimation on sample data. Supposing one has pairs of data as defined in Equation 2.1 comprising a set of observations x_n where $n = 1, \ldots, N$ together with corresponding target values t_n. The least-squares method estimates the model-parameters $\boldsymbol{w}$ such that the sum of the squares of the differences between the targets t_i and the straight line or plane respectively is a minimum. [Montgomery and Peck, 1992] covers the detailed mathematical derivation of the least-squares model-parameter estimation method.

Model adequacy checking. Once the model-parameters are found and the model is created, the fitting quality of the model has to be evaluated. Such analyses determine the usefulness of the regression model. An outcome of this adequacy checking indicates either a reasonable model or signifies that the model must be modified.

Most of the time the regression equation is only an approximation of the true relationship between the parameters and the data. Figure 2.2(a) sketches a relatively complex function which is very well approximated by *Linear Regression*. A *Linear Regression* not always leads to satisfying results. One sometimes investigates so-called piecewise linear regression models shown in Figure 2.2(b) [Montgomery and Peck, 1992].



(a) *LR* approximation (b) *LR* piecewise approximation

Figure 2.2.: Examples for *Linear Regression* approximation and piecewise *Linear Regression* approximation adapted from [Montgomery and Peck, 1992]. (a) shows a relatively complex function very well fitted by *Linear Regression* and (b) depicts a piecewise *Linear Regression* approach sometimes used when basic *Linear Regression* does not lead to a satisfying result.

Finally, the importance of data set quality should be taken into account. The drawbacks of either *Linear Regression* or *Multiple Linear Regression* analyses are conditional on the data set. Good data sets usually lead to simplified analyses and a more generally applicable model. In conclusion, representative data collection of the system studied is important to end up with proper results.

2.1.4. Support Vector Regression

Generally, the support vector algorithm or *support vector machine (SVM)* is based on the research done by Vapnik and Chervonenkis in 1964 in Russia. The *SVM* was invented by Vladimir Vapnik while the current standard *soft margin* approach was proposed by [Cortes and Vapnik, 1995]. *SVM* are supervised learning methods to analyse data and recognize patterns. This approach is currently used for classification as well as for regression analyses as already stated earlier in Section 2.1.2.

Theoretical Principle. As for all supervised learning methods first one needs to train the *SVM* using a training data set ahead of applying the *SVM* to the test data set. The interesting application of *support vector machines* for our task is *Support Vector Regression (SVR)*.

In *SVR* the basic idea is to find a function $f(x)$ that has at most ϵ deviation from the actually obtained targets for all training data, and at the same time is as smooth as possible [Smola and Schölkopf, 2004]. In other words one does not care about errors as long as they are in a certain ϵ-range, but deviations larger then ϵ are not permitted.

Consider for example a linear function

$$f(\boldsymbol{x}) = \sum_{i=1}^{D} w_i x_i + b. \tag{2.6}$$

We want to find a $\boldsymbol{w}$ such that for a training set denoted in Equation 2.1 all errors are at most of size ϵ, i.e., $|f(\boldsymbol{x_i} - t_i)| \leq \epsilon$. To achieve good generalization, it is known that small parameter vectors should be preferred. We therefore want to achieve this with a parameter vector $\boldsymbol{w}$ with minimal norm $\|\boldsymbol{w}\|^2$. In other words, we generally want to solve the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & |f(\boldsymbol{x_i}) - t_i| \leq \epsilon. \end{aligned} \tag{2.7}$$

Clearly, it will not always be possible to guarantee a maximum error of size ϵ. Therefore, instead of hard constraints, one tries to minimize a *soft margin loss function* which penalizes errors linearly that exceed ϵ. Analogously to the *soft margin loss function* some slack variables ξ_i, ξ_i^* can be introduced to provide the ability for the optimization problem stated in Equation 2.7 for errors exceeding ϵ. Thus, the formulation can be rewritten like

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{subject to} \quad & \begin{cases} f(\boldsymbol{x_i}) - t_i & \leq \epsilon + \xi_i^* \\ t_i - f(\boldsymbol{x_i}) & \leq \epsilon + \xi_i \end{cases} \end{aligned} \tag{2.8}$$

for $\xi_i, \xi_i^* \geq 0$ and $C > 0$. As stated in [Smola and Schölkopf, 2004] the soft margin parameter C symbolizes the trade-off between the amount up to which deviation larger than ϵ are tolerated and the flatness of the function.

Figure 2.3 shows the described *soft margin loss function* graphically. Points inside the shaded area are accepted as they are whereas points outside the shaded area with a larger deviation than ϵ are penalized in a linear fashion using the *soft margin loss function* shown on the right hand side. For instance point ζ on the left hand side gets linearly penalized as shown on the right hand side.

Dual optimization problem. The optimization problem described in Equation 2.8 can be solved in most cases in its dual formulation. Note that this is true if the dimensionality of the parameter vector $\boldsymbol{w}$ is much higher than the number of observations. The key idea of this dual formulation is to construct a Lagrange function by introducing a dual set of variables. Solving the Lagrange function as described in [Smola and Schölkopf, 2004]

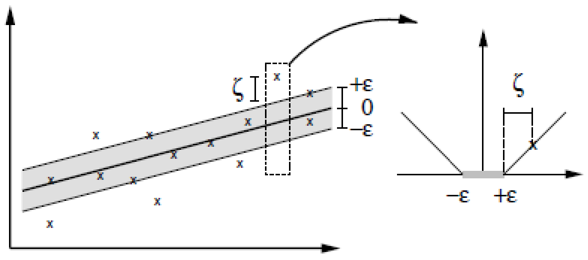


Figure 2.3.: Functionality of the *soft margin loss function* for a linear *SVM*. Points inside the shaded area are accepted whereas points outside the area get linearly penalized using the *soft margin loss function* on the right hand side [Smola and Schölkopf, 2004].

and [Chang and Lin, 2011] yields to the following dual optimization problem

$$\begin{aligned} \underset{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}{\text{minimize}} \quad & \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T \boldsymbol{x}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \epsilon\sum_{i=1}^{N}(\alpha_i+\alpha_i^*) + \sum_{i=1}^{N} t_i(\alpha_i-\alpha_i^*) \\ \text{subject to} \quad & \boldsymbol{e}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \end{aligned} \tag{2.9}$$

where $0 \le \alpha_i, \alpha_i^* \le C$ and $i = 1, \ldots, N$. Solving the dual optimization problem in Equation 2.9 one ends up with the so-called *support vector expansion*. Equation 2.10 shows this approximation function

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b \tag{2.10}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product, which bases on the explicit solution for w

$$w = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)x_i. \tag{2.11}$$

Analysing Equation 2.11 one can see that w can be completely described as a linear combination of the training patterns x_i. Moreover, the algorithm can be described in terms of dot products between the data. Thus, in order to calculate $f(x)$ it is not necessary to compute w explicitly which will come in handy for the non-linear formulation of *SVR* [Smola and Schölkopf, 2004].

Computing parameter b. Before making the algorithm non-linear the calculation of the missing parameter b in Equation 2.10 will be covered briefly. The most famous solution to obtain b are the so-called Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions predicate that the product between the dual variables and the constraints has to vanish at the point of the solution. Knowing these fact allows one to make several conclusions. These conclusions and further details about the computation of parameter b are shown in [Smola and Schölkopf, 2004].

Non-linear mapping. The next step is to make the *SVM* non-linear in order to apply the algorithm to non-linear problems. This can be achieved by pre-processing the training patterns $\boldsymbol{x}_i$ by a map $\phi : \mathcal{X} \rightarrow \mathcal{F}$ into some high dimensional feature space $\mathcal{F}$ and use a linear model in feature space. In fact, the algorithm can be rewritten such that the transformation $\phi(x)$ does not have to be computed explicitly. Instead, one only needs to compute products in feature space implicitly [Smola and Schölkopf, 2004].
Hence, it suffices to know

$$K(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle \tag{2.12}$$

rather than ϕ explicitly where $\langle \cdot, \cdot \rangle$ denotes the inner product. K is called the kernel. This so-called *kernel trick* is needed in order to make the feature expansion computationally feasible [Gunn, 1998]. One popular example for non-linear modelling is provided by the polynomial mapping using a kernel like

$$K(x_i, x_j) := \langle x_i, x_j \rangle^d \tag{2.13}$$

where d denotes the degree of the polynomial. Other kernels are Gaussian or exponential radial basis functions (RBF) as well as Fourier Series or Splines.

Rewriting the support vector expansion shown in Equation 2.10 for the non-linear approach one results in

$$f(x) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{2.14}$$

Once the *SVR* is trained well on the training data set, one is able to apply it to the test data set and evaluates a prediction based on the model-parameters learned during the training-phase.

***SVR* properties.** The main advantages of *SVM* are

- finding the optimal separation hyperplane
- dealing with very high dimensional data
- less input hyper-parameter during training
- usually work very well

whereas the disadvantages can be addressed like

- require both positive and negative examples
- need to select a good kernel function
- require lots of memory and CPU time.

All together, *SVM* and in detail *SVR* is a technique worth trying out to solve the task we are working on.

2.1.5. Gaussian Processes

Theoretical Principle. A *GP* is used to describe a probability distribution of functions. These probability distribution is specified by the distribution of any two arbitrary points in a function. In case of *GP* the variance of these two arbitrary points are normally distributed. For instance, the two arbitrary data points can be normally distributed in dependency of the distance between the two points as shown in Equation 2.17. Thus, adjacent points are similar whereas the variance is big for two points located far from each other. For this reason, the covariance of two points close to each other is bigger than the one of two points located far from each other.

Formally, [Rasmussen and Williams, 2006] defines a *GP* as follows:

Definition 2.1.1. *A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.* □

In fact, a *GP* is completely specified by its mean and its covariance function, which is one of the main properties of *GP*. The mean function $m(\boldsymbol{x})$ and the covariance function $cov(\boldsymbol{x}, \boldsymbol{x'})$ of a real process $f(\boldsymbol{x})$ are given as

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}\big[f(\boldsymbol{x})\big] \\ cov(\boldsymbol{x}, \boldsymbol{x'}) &= \mathbb{E}\big[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'}))\big] \end{aligned} \tag{2.15}$$

where $cov(\boldsymbol{x}, \boldsymbol{x'}) = k(\boldsymbol{x}, \boldsymbol{x}')$ and $k(\boldsymbol{x}, \boldsymbol{x}')$ is known as the kernel function in literature [Rasmussen and Williams, 2006], [Bishop, 2007].

Formally, a *Gaussian Process* will be written as

$$f(\boldsymbol{x}) \sim \mathcal{GP}\big(m(\boldsymbol{x}), cov(\boldsymbol{x}, \boldsymbol{x'})\big). \tag{2.16}$$

Most applications do not provide prior knowledge about the mean $m(\boldsymbol{x})$ and so by symmetry it is common to set it to zero.

In general, a kernel $k(x, x')$ describes variance behaviour for arbitrary points x and x'. A kernel function can be defined directly. One possible choice is the Gaussian kernel defined as

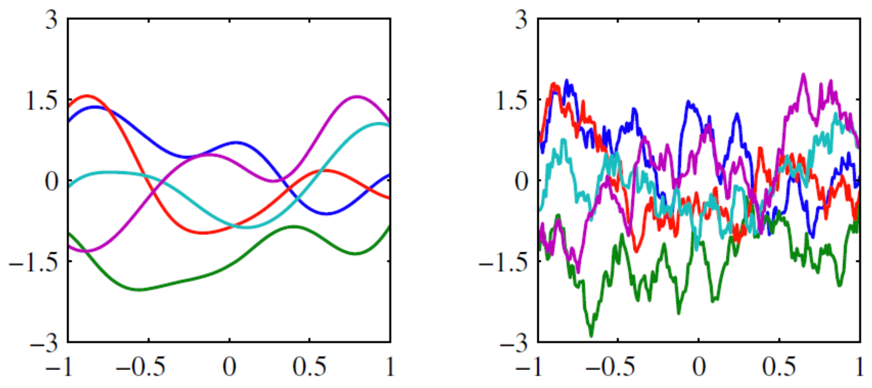
$$k(\boldsymbol{x}, \boldsymbol{x'}) = exp\left(-\frac{||(\boldsymbol{x} - \boldsymbol{x'})||^2}{2\sigma^2}\right) \tag{2.17}$$

while another choice can be the exponential kernel given by

$$k(x, x') = exp\left(-\theta|(x - x')|\right). \tag{2.18}$$

σ and θ in Equation 2.17 and 2.18 are so-called hyper-parameters, described in detail later on, defining the shape of the kernel.

Figure 2.4 shows samples of functions drawn from two *GP*s with different kernels. Caused by the shape of the kernel function the Gaussian kernel in Figure 2.4(a) appears to smoother whereas the exponential kernel Figure 2.4(b) drops and rises faster and steeper, thus it is fluctuating more compared to a Gaussian kernel.



(a) Gaussian kernel (b) Exponential kernel

Figure 2.4.: Samples from *GP* with different kernel functions. The Gaussian kernel causes smoother behaviour in (a) whereas the exponential kernel leads to a more fluctuating shape in (b) (Source: [Bishop, 2007]).

Application of *Gaussian Processes*. *GP* can be used to solve both supervised learning tasks, either classification or regression problems introduced in section 2.1.2.

However, considering our task only the regression application of *GP* draws our attention. Hence, the classification task will be skipped and in the following the regression will be reflected upon briefly.

Gaussian Processes for regression. In *Gaussian Process* regression tasks one esteems a set of functions to be probable. Using the training data and the set of functions the hyper-parameters defining the kernel functions can be learned. These kernel function define the appearance of the distribution. In other words, based on the trainings data and the set of functions assumed to be probable some supporting points of the function can be found.

In general, as introduced in [Rasmussen and Williams, 2006] *GP* regression can be viewed from two ways. On the one hand, the weight-space view and on the other hand the function-space view. In this thesis the function-space view is applied.

For modelling more realistic models one does not have access to function values directly, but only to their noisy versions formally noted as

$$t_n = y_n + \epsilon_n \tag{2.19}$$

where ϵ_n is a random noise variable independently chosen for each observation and $y_n = f(x_n)$. Considering the noise process having a Gaussian distribution leads to an isotropic Gaussian of the form

$$p(\boldsymbol{t}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{t}|\boldsymbol{y}, \beta^{-1}\boldsymbol{I}_N) \tag{2.20}$$

where $\boldsymbol{I}_N$ denotes a $N \times N$ unit matrix. β represents the precision of the noise.

As shown in [Bishop, 2007] the marginal distribution $p(\boldsymbol{t})$ can be found by integrating over $\boldsymbol{y}$. This leads to

$$p(\boldsymbol{t}) = \int p(\boldsymbol{t}|\boldsymbol{y})p(\boldsymbol{y})d\boldsymbol{y} = \mathcal{N}(\boldsymbol{t}|\boldsymbol{0}, \boldsymbol{C}) \tag{2.21}$$

where the mean is set to zero and the covariance matrix is defined as

$$C(x_n, x_m) = k(x_n, x_m) + \beta^{-1}\delta_{nm} \tag{2.22}$$

where δ_{nm} denotes a unit vector with the restriction that the kernel matrix $k(x_n, x_m)$ must be positive semi-definite.

The goal in regression is to consider new inputs (test points) and make proper predictions of the target values. Based on the learned shape of the *GP* the function value of a test point can be non-linearly interpolated or the distribution can be predicted. More formally, supposing a training set comprising of $\boldsymbol{t}_N = (t_1, \ldots, t_N)^T$ corresponding to input values $\boldsymbol{x}_N = (x_1, \ldots, x_N)^T$our goal is to predict the target variable t_{N+1} for a new input vector $\boldsymbol{x}_{N+1}$. In order to do so we are interested in $p(t_{N+1}|\boldsymbol{t}_N)$. From 2.21 the joint distribution over $t_1, \ldots, t_{N+1}$ is given by

$$p(\boldsymbol{t}_{N+1}) = \mathcal{N}(\boldsymbol{t}_{N+1}|\boldsymbol{0}, \boldsymbol{C}_{N+1}) \tag{2.23}$$

with the covariance matrix $\boldsymbol{C}_{N+1}$ defined as

$$\boldsymbol{C}_{N+1} = \begin{pmatrix} \boldsymbol{C}_N & \boldsymbol{k} \\ \boldsymbol{k} & c \end{pmatrix} \tag{2.24}$$

where $\boldsymbol{C}_N$ is the $N \times N$ covariance matrix with elements stated in Equation 2.22 and the vector $\boldsymbol{k} = k(\boldsymbol{x}_n, \boldsymbol{x}_{N+1})$ for $n, m = 1, \ldots, N$. Finally, the scalar c represents the newly added input as

$$c = k(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1}) + \beta^{-1}. \tag{2.25}$$

Putting everything together as shown in [Bishop, 2007] one can see that the conditional distribution $p(t_{N+1}|\boldsymbol{t})$ can be seen as a Gaussian distribution with mean and covariance given by

$$\begin{aligned} m(\boldsymbol{x}_{N+1}) &= \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{t} \\ \sigma^2(\boldsymbol{x}_{N+1}) &= c - \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{t} \end{aligned} \tag{2.26}$$

where $\boldsymbol{C}_N$ is the $N \times N$ covariance matrix and $\boldsymbol{k}$ is the kernel vector $\boldsymbol{k} = k(\boldsymbol{x}_n, \boldsymbol{x}_{N+1})$ for $n = 1, \ldots, N$. Vector $\boldsymbol{t}$ represents the training set of noisy function values of size N. Finally, the scalar c represents the newly added input. These equations completely define *GP* for regression.

Figure 2.5 shows an example of *GP* regression applied to a sinusoidal data set. The green curve represents the sinus from which the data points in blue are drawn by sampling and adding Gaussian noise. The mean of the *GP* predictive distribution is reflected by the red line while the shaded region corresponds to $\pm$ two standard deviations. Moreover, one recognizes the increasing uncertainty for the very right of the data points.

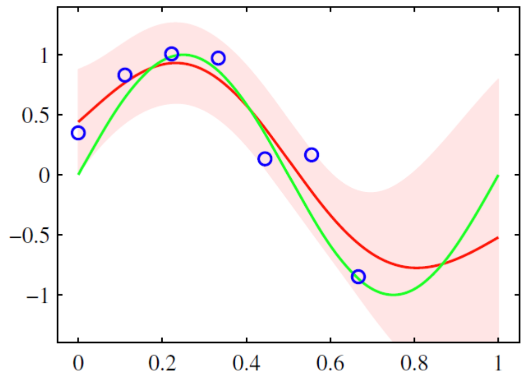


Figure 2.5.: *GP* regression example applied on a sinusoidal data set. (Source: [Bishop, 2007]). Green curve shows the sinusoidal function while blue points are the obtained data points. The red line shows the mean of the *GP* and the shaded area corresponds to plus and minus two standard deviations.

Learning the hyper-parameters. The prediction quality of a *GP* depends on the choice of the covariance function. Typically the covariance function will have some free parameters. Usually, free parameters will be called hyper-parameters and denoted as θ in this thesis [Rasmussen and Williams, 2006]. The hyper-parameters can be either fixed or learned by using a family of parametric functions and inferring the hyper-parameter values from the data. Length scale of the correlations and noise precision are governed by these hyper-parameters which correspond to the hyper-parameters in a standard parametric model. A common technique to learn hyper-parameters is the log likelihood maximization like

$$\frac{\partial}{\partial \theta_i} ln\ p(\boldsymbol{t}|\boldsymbol{\theta}) = -\frac{1}{2} Tr \left(\boldsymbol{C}_N^{-1} \frac{\partial \boldsymbol{C}_N}{\partial \theta_i]} \right) + \frac{1}{2} \boldsymbol{t}^T \boldsymbol{C}_N^{-1} \frac{\partial \boldsymbol{C}_N}{\partial \theta_i]} \boldsymbol{C}_N^{-1} \boldsymbol{t} \tag{2.27}$$

where $\boldsymbol{C}_N$ is the $N \times N$ covariance matrix with elements stated in Equation 2.22 and Tr denotes the trace of a matrix formally defined as $Tr(\boldsymbol{A}) = \sum_{i=1}^{N} a_{ii}$. This can be done efficiently by applying optimization algorithms such as conjugate gradients (CG) [Bishop, 2007].

From a computational point of view the inversion of matrix $\boldsymbol{C}$ is very intensive and requires $\mathcal{O}(N^3)$ computations. In fact, this is a large drawback particularly for the application on large data sets. However, the computational costs drop to $\mathcal{O}(N^2)$ once the inversion of the covariance matrix $\boldsymbol{C}$ is known [Farrell and Correa, 2007].

Detailed influences of the hyper-parameter choice can be found in Chapter 2.3 in [Rasmussen and Williams, 2006].

***GP* automatic relevance determination (*GP ARD*).** [Rasmussen and Williams, 2006] extended the hyper-parameter learning approach mentioned in the previous section by introducing a separate hyper-parameter for each input variable. As a result, the optimization of the hyper-parameters by maximum likelihood allows a inference to the relative importance of different inputs.

For simplicity, consider a *GP* with a two-dimensional input space and the following kernel function

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \theta_0 \exp \left(-\frac{1}{2} \sum_{i=1}^{2} \eta_i (x_i - x_i')^2 \right) \tag{2.28}$$

where η_i denote the newly added hyper-parameters. Adapting these η-values to the data set using maximum likelihood it becomes possible to detect whether an input value has little effect on the predictive distribution result. This can be useful to discard such inputs.

Figure 2.6 represents the η-values of a simple synthetic data set consisting of three inputs x_1, x_2 and x_3. 100 values of each function are sampled to generate the target variable t. x_1 and x_2, sampled from the same Gaussian distribution, only differ from each other by the addition of different noise whereas x_3 is sampled from a independent Gaussian distribution. However, the *GP* with *ARD* optimizes its marginal likelihood using scaled conjugate gradients algorithm. Figure 2.6 plots the hyper-parameters η_1, η_2 and η_3 as a function of the number of iterations. Note the logarithmic scaling of the vertical axis. Interpreting the results depicted in Figure 2.6 one can see that η_1 (red) performs slightly better than η_2 (green) whereas η_3 (blue) becomes very small indicating the irrelevance of x_3 to the prediction of the target t.

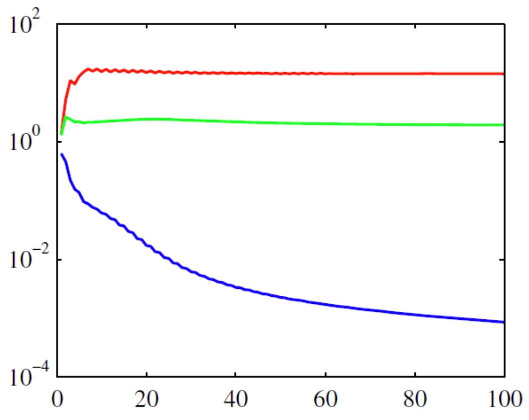


Figure 2.6.: *Automatic relevance determination* in a *GP*. Three inputs x_1, x_2 and x_3 are represented through their corresponding hyper-parameters η_1 in red, η_2 in green and η_3 in blue as a function of the number of marginal likelihood optimization iterations. η_1 performs slightly better than η_2 where both outperform η_3 which indicates its irrelevance to the target prediction (Source: [Bishop, 2007]).

Finally, [Bishop, 2007] proposed the following form of kernel function including ARD which is evaluated to perform properly for *GP* applications on regression tasks

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \theta_0 \exp\left(-\frac{1}{2}\sum_{i=1}^{D}\eta_i(x_{ni} - x_{mi})^2\right) + \theta_2 + \theta_3 \sum_{i=1}^{D} x_{ni}x_{mi}. \tag{2.29}$$

where D denotes the dimensionality of the input space [Bishop, 2007].

2.2. Applications

2.2.1. Linear Regression

The application of *Linear Regression* is a very basic and essential method. Thus, no specific applications are considered. As [Montgomery and Peck, 1992] mentioned they occur in almost every field including

- engineering
- physical sciences
- economics
- life and biological sciences
- management

- social sciences.

Uses of Regression. Regression models in general are useful for several purposes like

- data description
- parameter estimation
- prediction and estimation
- control.

[Bishop, 2007] and [Montgomery and Peck, 1992] provide a large amount of applications for either *Linear Regression* or *Multiple Linear Regression*.

2.2.2. *Support Vector Regression*

A very common application of *SVM* addressing both classification and regression problems is *face recognition*. [Li et al., 2000] covered a *multi-view face detection and recognition*. Additionally, *financial forecast* is another domain of application of the *SVR* approach. [Trafalis and Ince, 2000] investigated the use of *SVR* to forecast stock prices and compared them with other techniques like back-propagation and RBF networks. Similarly, [Tay and Cao, 2001] covered the topic of financial time series forecasting with *SVM* and compared them with *multi-layer back-propagation neural networks*. An even better approach in terms of time series forecasting is proposed by [Lu et al., 2009]. They get rid of one of the key problems in modelling financial time series, which is having inherent high noise by combining *independent component analysis (ICA)* and *SVR*. They used *ICA* as pre-processing step of the *SVR* to remove the noise and thus improved their results considerably compared to conventional *SVR* approaches. However, even completely different topics already made use of the *SVM* approach. For instance, [Yuan and Huang, 2004] used *SVR* to predict important properties of proteins, particularly *protein accessible surface areas (ASA)*. Moreover, *SVR* is also used in applications of iron works where our task belongs to. As an example, [Wen et al., 2009] investigated corrosion rate prediction of 3C steel under different seawater environment by using *SVR*. In the following, few approaches and their application will be explained briefly.

Face Recognition. *SVM* are able to deal well with *face recognition* problems. For instance, [Li et al., 2000] considered a multi-view face detection and recognition framework. They constructed several detectors, each of them in charge of one specific view. Generally speaking, its more probable having one person in different poses than many persons in the same pose. However, if the pose of a face image is known, the recognition problem can be simplified to a great extent. Thus, they used a *SVR* approach to estimate the pose followed by a support vector classification to identify the face.

In fact, two pose estimators are trained to estimate the yaw angles, on the one hand and the tilt angles of the image faces on the other hand. The training set consisted of 1956 face images of size 20×20 pixel taken from 12 subjects where one image per pose containing poses from 0° - 180° in yaw and 60° - 120° in tilt with steps of 10°. A Gaussian kernel is used. Firstly, all images are pre-processed applying sobel filters and principal component analysis (PCA) to reduce the dimension as well as some simple normalisation is done. Then, these patterns are fed to the training algorithm. The generalisation performance of the trained pose estimators are tested using another set of 1283 face images. This results shows that low dimensional *PCA* representation (20-30 in dimension per example) can provide a satisfactory performance in pose estimation. The estimated results of the pose estimators are fed into a set of *SVC* classifiers to perform the *face recognition*. Finally, all these issues are integrated into a *SVM*-based framework. Looking at the test results, [Li et al., 2000] obtained a detection rate above 95% and a recognition accuracy above 95%. The pose estimation in both yaw and tilt is around 10°. All these results are collected while testing their framework on four live video sequences containing faces and across views.

In conclusion, one can see *SVR* works well to solve *face recognition* tasks.

Tourism Demand Prediction. Tourism industry has emerged as the fastest growing sector all over the world in the past few decades. Tourism expenditure has become an important source of economic activity, employment, tax revenue, income and foreign exchange. Hence, every country has to understand its international visitors. Moreover, unstockpiled economics such as empty hotel rooms or air flight seats should be avoided. Thus, accurate forecasting of tourism demand, both the short and the long term becomes an more and more essential requirement. [Chen and Wang, 2007] postulated a *SVR* based approach to predict the tourism demand. As a matter of fact, the hyper-parameters of a *SVR* has to be set carefully in order to achieve an efficient and appropriate forecast. Inappropriate hyper-parameters in *SVR* lead to *overfitting* or *underfitting*. Among others, these hyper-parameters include the kernel function as well as the bandwidth of the kernel function. However, no general guidelines are available to select these hyper-parameters. The generalization performance does not only depend on one rather than on the interaction of all hyper-parameters. Therefore they investigated a *real-value genetic algorithm (RGA)*, known as *GA-SVR* to determine hyper-parameters of the *SVR*. Apart from known techniques *GA-SVR* is no trial-and-error procedure. These algorithm simultaneously optimizes all *SVR* hyper-parameter from the training data. Skipping the detailed explanation, this algorithm leads to the optimal hyper-parameters in the end. Having the important hyper-parameters they created a experimental setting for forecasting an univariate time series. The past, lagged observations of the time series are the input and the future values the output of the *SVR*. As input data set, tourism data of China between 1985 and 2001 is used. 64 input patterns are used, where the prior 56 input patterns are employed for the training sample to model the *SVR*. The remaining 8 input patterns are used for testing samples to estimate the forecasting capacity of the *SVR*. The result of the trained *SVR* is compared with *back-propagation neural networks (BPNN)* and *autoregressive integrated moving average models (ARIMA)*.

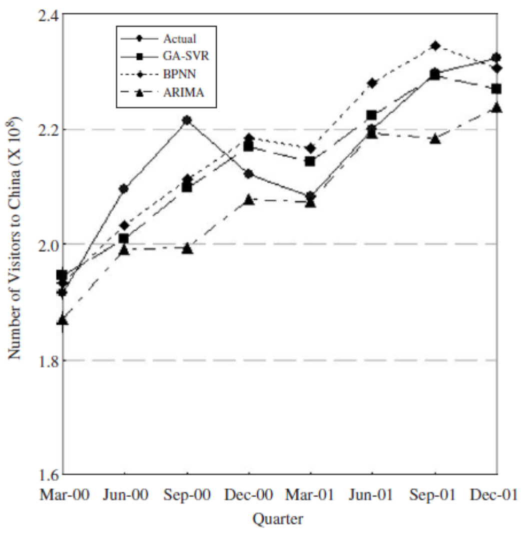


Figure 2.7.: Performance of different tourism demand forecasting methods from [Chen and Wang, 2007]. All models lead to appropriate results. The *GA-SVR* outperforms *BPNN* and *ARIMA* in terms of forecasting the tourism demand.

Looking at Figure 2.7 one can see that the forecasting performance of all three models are appropriate. In detail, *GA-SVR* models are superior to those from the other two models. In other words, the *GA-SVR* outperformed *BPNN* and *ARIMA* in terms of forecasting the tourism demand.

In conclusion, combining *genetic algorithms (GA)* and *SVR* result in a perfect forecasting approach for non-linear time series. Within the forecasting fields of tourism demand, the *GA-SVR* is a reliable forecasting tool.

Prediction of corrosion rate of 3C steel under different seawater environments. [Wen et al., 2009] used *SVR* to predict the corrosion rate of 3C steel which is a carbon steel under different seawater environments. In fact, the increasing tonnages of carbon steel, dramatically attracting people's attention to the question of corrosion. Seawater is recognized to be one of the most corrosive natural electrolytes under natural environment. They investigated a corrosion prediction using *SVR* combined with a technique called *particle swarm optimization (PSO)* to optimize the hyper-parameters. Finally, the results are compared to *BPNN* results. The data set contains 46 samples from different seawater environments measured by electrochemical techniques. Five attributes

- temperature
- dissolved oxygen
- salinity
- pH value
- oxidation-reduction potential

are employed as input variables while corrosion rate acted as output variable. Five out of 46 samples are used as test samples accompanied with 41 training samples. The *SVR* model as well as the *BPNN* is trained with the same data sets to provide a perfectly suited test environment for comparing reasons.

As a result, looking at Table 2.1 it can be seen that the maximum absolute percentage errors predicted by *SVR* are bigger than those of BPNN. However, the majority of samples have smaller predicted percentage errors determined by *SVR* models compared to *BPNN*. Thus, the prediction accuracy of *SVR* is greater than *BPNN*. We note that an increase of the amount of training samples can contribute to improved regression accuracy.

No.	**Experimental**	**BPNN**		**SVR**	
	rate	Pred. rate (μAcm^{-2})	Rel. error (%)	Pred. rate (μAcm^{-2})	Rel. error (%)
7	14.06	13.12	-6.69	14.195	0.96
10	17.11	17.14	0.175	16.786	-1.88
14	10.578	11.1	4.93	10.928	3.31
19	11.446	12.08	5.54	11.653	1.81
21	12.553	11.59	-7.67	13.963	11.23

Table 2.1.: Comparison between electrochemical measurement values and estimated results by using *BPNN* and *SVR* methods, respectively, for five test samples taken from [Wen et al., 2009].

In conclusion, *SVR* is able to predict corrosion rate of carbon steel under 5 different seawater environments very satisfying and is therefore applicable to iron works as well. In detail, the predicted errors of *SVR* are than those of *BPNN* using identical training and test data sets. Finally, even the generalization ability of the *SVR* model is superior to that of *BPNN*.

2.2.3. *Gaussian Processes*

As a matter of fact, *GP* are a very recent and state of the art approach. For this reason, the amount of former researches is manageable. Nevertheless few past researches exist. To be honest, *GP* is a very good tool dealing with non-linear data sets.

Considering energy generation, one of the most recent topics is renewable energy. [Mori and Kurata, 2008] investigated in the forecast of wind speed using *GP*, which is essential for wind power generation. Another closely related topic covered in the past is the air temperature prediction proposed by [Zhang et al., 2011]. Each time a new approach arises estimations or recognitions on human faces is a famous research domain. Thus, [Zhang and Yeung, 2010] drew their attention on personalized age estimation using *GP*. Moreover, *Gaussian Processes* will be used in completely different topics like prediction of financial stock trends by [Farrell and Correa, 2007] as well.

Few applications will be introduced briefly in the following.

Wind speed forecast. Renewable energy becomes more and more attractive to protect our environment. Though, the uncertainty of wind power output makes it quite difficult to deal with wind power generation. Logically, wind speed remarkably effects the power generation output. Thus, big market players are interested in short term wind speed prediction.

[Mori and Kurata, 2008] used *GP* to estimate the upper and lower bound of wind speed as well as the average. The authors used data from May to July 2006 measured at the Muroto Cape in Japan split into 144 training data and 24 test data parts to predict the wind speed one hour ahead. Station pressure, sea-level pressure, average temperature, humidity, rainfall level and wind speed are investigated as input each at time t to predict the output wind speed at time $t+1$. Their results are compared with results achieved by a multi-layer perceptron (MLP) and by RBF, both used as an artificial neural network (ANN) .

Looking at Figure 2.8 one can see that *GP* performs better than MLP and RBF in both average and maximum error.

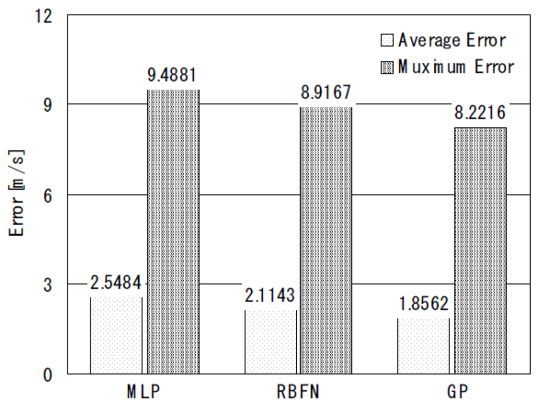


Figure 2.8.: Error comparison of one-hour ahead wind speed prediction using different approaches taken from [Mori and Kurata, 2008]. Looking at both the maximum and the average error the *GP* performs better than the RBF network and the multi-layer perceptron in terms of one-hour ahead wind speed prediction.

As a result, *GP* performs better than former approaches on short term wind speed prediction.

Air temperature prediction. Weather is in fact an environment variable which has huge impact to humans life. Therefore, weather forecasts, which are classic non-linear problems, are research topics always gaining lots of interest. One key player in weather forecast is the prediction of the air temperature.

[Zhang et al., 2011] used a combination of generalized partial least squares (GPLS) algorithm and the *GP* approach to predict the air temperature. They combined these two algorithms to take the good advantages of both approaches to reach higher forecast performance. Measurements of 150 days in Izmir, Turkey are used to apply their proposed GPLS-GP algorithm. $2/3$ of the data set act as training set while the remaining $1/3$ is used to test their algorithm. Among others, [Zhang et al., 2011] used the root mean squared error (RMSE) to evaluate their performance. Conventional methods like partial least squares (PLS) and generalized partial least squares (GPLS) are used for comparison.

Looking at the RMSE in Table 2.2 as well as the prediction deviation in Figure 2.9 one can see that GPLS-GP approach produces the smallest prediction error indicating the best performance of the compared methods. Especially Figure 2.9 shows increasing errors at the very right of the plot for conventional approaches.

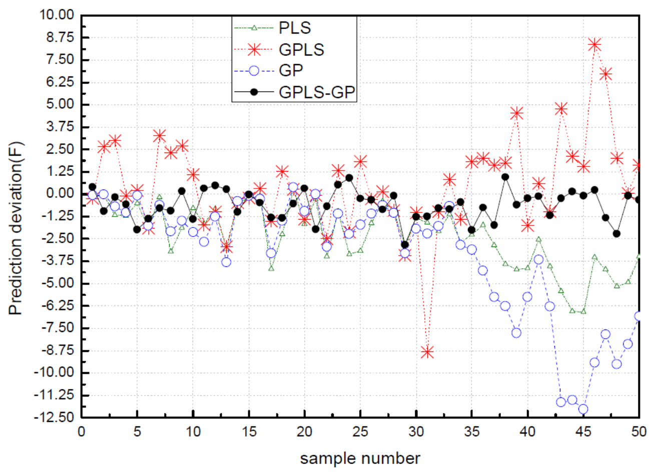


Figure 2.9.: Prediction error [F°] versus sample number of the compared methods PLS, GPLS, GP and GPLS-GP taken from [Zhang et al., 2011]. The GPLS-GP leads to the best prediction accuracy indicated through the smallest deviation error. Other conventional approaches perform worse and tend to increased prediction errors, especially for higher sample numbers.

Concluding the research of [Zhang et al., 2011] results are improved in terms of robustness and accuracy notably.

	PLS	GPLS	GP	GPLS-GP
pred. RMSE [F°]	2.9031	2.7000	4.7365	**1.0218**

Table 2.2.: Predicted RMSE comparison of different methods [Zhang et al., 2011].

Age estimation. Identification or recognition methods based on human face images are one of the most interested and covered tasks in computer vision. Particularly, computer vision applications are interested in automatic age estimation. Facial images contain rich information about a person, e.g. identity, gender, expression, age, etc. Recent researches show that personalized age estimation methods basically outperform global age estimation algorithms.

[Zhang and Yeung, 2010] proposed a novel personalized age estimation approach by formulating the task as a multi-task learning problem. In detail, a special form of *GP* called warped *GP* is used as a basis to propose the so-called *multi-task warped Gaussian processes (MTWGP)* . However, multi-task learning in general is a learning paradigm focussing on the generalization performance of the task with the help of some other related tasks. Thus, each person is treated as a separate task and each task is estimated using a warped *GP* (WGP) explained by [Zhang and Yeung, 2010] in detail.

Considering age estimation, some common pattern are shared by all face images. For instance, the whole face or even facial parts will become bigger as a person grows from a child to an adult. Moreover, facial wrinkles are a common pattern usually increasing as one gets older. [Zhang and Yeung, 2010] distinguishes facial features of the ageing process in two parts. We have, on the one hand, features common to all persons and on the other hand person specific features. Thus, a multi-task regression estimator is proposed depending on both common features as well as features learned separately for each person using MTWGP.

[Zhang and Yeung, 2010] tested their approach with the help of two age databases FG-NET[1] and MORPH [Ricanek and Tesafaye, 2006]. The FG-NET contains 1002 facial images consisting of 6-18 images of 82 different persons varying in age from 0 to 69. MORPH database comprises 1724 facial images from 515 persons with only 3 images per person. Persons in MORPH database vary in age from 15 to 68. To examine MTWGP performance the mean absolute error (MAE) as well as the cumulative score (CumScore) are compared to former results. Having t test images, one supposes $t_{e\leq l}$ images have a absolute prediction error smaller than l. Then, the CumScore at error level l can be calculated like

$$CumScore(l) = t_{e\leq l}/t \times 100\%. \tag{2.30}$$

In [Zhang and Yeung, 2010] only cumulative scores smaller than $l = 10$ years are treated

[1]http://www.fgnet.rsunit.com

as acceptable.

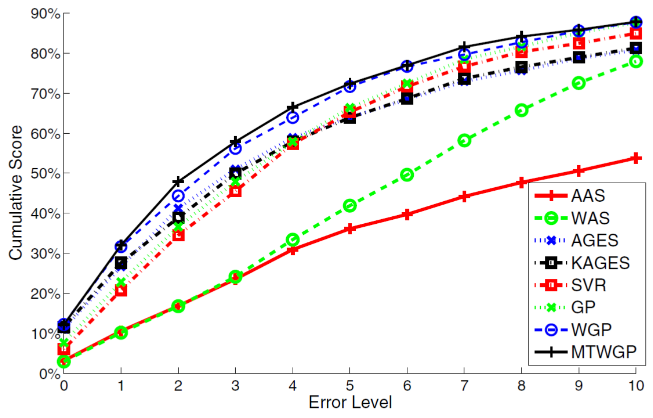


Figure 2.10.: Cumulative scores at error levels from 0 to 10 of different ages estimation methods applied on facial images from the FG-NET database taken from [Zhang et al., 2011]. MTWGP leads to the best results in the domain of personalized age estimation. This plot shows that other state-of-the-art approaches can not achieve comparable results.

As a result, Figure 2.10 shows that MTWGP even outperforms other state-of-the-art approaches in the domain of personalized age estimation. Similar results for the MORPH database are shown in [Zhang et al., 2011].

Financial stock trend prediction. [Farrell and Correa, 2007] use *GP* regression to predict stock trends. They examined two different experiments. Firstly, past stock prices are used to predict whether the price will rise or fall the next day. Secondly, the *GP* are investigated to predict the stock price explicitly.

A *GP* for stock prices can be seen as a function over time. [Farrell and Correa, 2007] investigated in different kernels defining the covariance like

- squared exponential
- matern class
- rational quadratic.

Given a kernel the optimal set of hyper-parameters is optimized using 100 iteration steps to fit the optimized data by maximizing the marginal likelihood as explained in Section 2.1.5.

From a practical point of view, 8 stocks are considered and their data is collected over a period of 10 years. As a result, the prediction of the up and down trend turns out to be highly sensitive to the amount of training data used. Good results can be achieved using a matern class covariance function trained on a data set of one month. The single-stock price prediction leads to more reliable results using long time training data sets. [Farrell and Correa, 2007] measured penalty in terms of regret. Having the next day (t+1) stock prices in advance they are able to find which stock would have been the best to invest the day before (t). The difference between the real price and the predicted price between day (t) and (t+1) is calculated. Wrong predictions lead the penalty to become the gain of the best stock.

Summing up, stock trend prediction is more useful on smaller data sets whereas single-stock price prediction performs well on long term predictions (bigger data sets). From the kernel-selection point of view matern class or squared exponential kernels lead to the best results shown in Table 2.3.

Time period	Squared exponential	Matern $\nu = 3/2$	Rational quadratic
10 day	Stock 7, 0.24 regret	Stock 7, 0.24 regret	no stock predicted
1 month	Stock 7, 0.24 regret	Stock 7, 0.24 regret	Stock 7, 0.24 regret
6 month	Stock 7, 0.24 regret	Stock 7, 0.24 regret	Stock 7, 0.24 regret
1 year	Stock 8, 0 regret	Stock 8, 0 regret	Stock 7, 0.24 regret
3 years	Stock 8, 0 regret	Stock 8, 0 regret	Stock 7, 0.24 regret

Table 2.3.: Predicted stock with regret in covariance function for different time periods taken from [Farrell and Correa, 2007].

In conclusion, *GP* provide a new and quite competitive method for further researches on non-linear data samples in various domains.

2.3. Conclusion

This chapter basically introduced the theoretical background of the approaches *LR*, *SVR* and *GP* used in APO. In addition, former researches using the same approaches in similar and completely different domains are covered briefly.

So far, we know the theoretical background and the principles used in this thesis. The next chapter focusses on their application and required adaptations in terms of APO.

Teil III.

APPROACH

3 Approach

Contents

This chapter introduces the approaches applied in this thesis. Section 3.1 gives a thesis overview and basically describes how the procedure to achieve the aimed results is structured. Section 3.2 briefly introduces the input data sets. Section 3.3 focusses on data pre-processing which actually is a very important part in order to result in good solutions. Finally, Section 3.4 focusses on APO specific simplifications of input data as on use of the supervised learning concepts adapted for our purposes.

3.1. Overview

As mentioned in Section 1.3, finding relationships between production process parameters and refractory wear is the topic of interest. For this reason, *Linear Regression* and *Multiple Linear Regression* analysis and non-linear regression analyses like *Support Vector Regression* as well as *Gaussian Processes* are applied. *SVR* and *GP* are used to construct a model representing the process. The main goal of these analyses is to achieve a sense-full model resulting in proper predictions of future behaviour of the converter. Particularly, the refractory lining wear in the analysed area should be predicted. In other words, one should be able to predict the evolution of areas inside the converter.

Recap, all analysis are based on a data set comprising

(1) laser measuring data of the remaining converter lining thickness

(2) production process data.

Maintenance data and refractory lining information will be added to APO in the future. These inputs are not included in this thesis. The general APO data flow with more detailed information about the input parameters is sketched in Figure 3.1. Once all input parameters are collected they are passed through a data-link connection to the master server. The master server performs the analyses of interest. In fact, this data flow is a future aim of APO. In this thesis all analyses are determined locally on a PC or notebook.

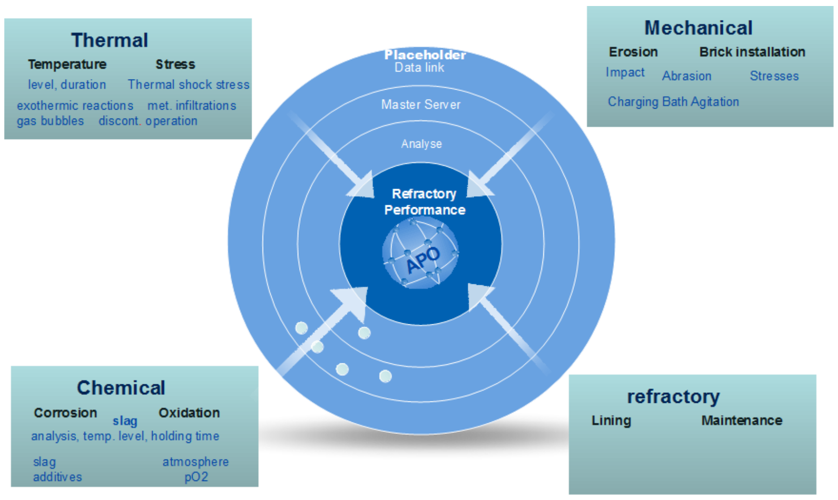


Figure 3.1.: Data flow of APO approach [Lammer, 2011]. In the beginning all data is collected ahead of passing it to the master server through a data-link connection. The analyses of interest are performed on the master server.

In advance to all analyses, the input data has to be pre-processed. Laser data has to be checked for inconsistency, missing values and some other criteria covered later in this chapter.

Summing up, the tasks covered in this thesis are

(1) data pre-processing

(2) finding relationships between production process parameters and refractory wear

(3) creating a model of the converter process using different approaches.

3.2. Data set

Basically, all input information is stored in laser measurement data on the one hand, and production process data on the other hand. This section introduces the format of the input files.

3.2.1. Laser measurement data

Generally speaking, each laser measurement produced by LaCam® consists of two files

(1) meta data file

(2) measurement data.

Both files are delivered in a raw ASCII text format with the extension `*.lch`.

Meta data file. The meta data file delivers information about the current converter status including date, time, campaign number, heat number etc. Listing 3.1 shows an excerpt of a typical laser meta data file produced by LaCam®. One row always contains an indicator and the corresponding value separated by a tabulator. For instance, row 12 contains the indicator `'Standreise'` corresponding to the campaign number and, separated by a tabulator, the value. Nevertheless, the meta data file contains lots of overhead. For this reason, this ASCII text file is pre-processed to fetch the data of interest as shown later in Section 4.2.

```
1  [...]
2  Stahlwerk   HKM
3  Stand   2
4  Zustellung  740501
5  Zust. Typ   Working
6  Posit. Code 5
7  Mess. Typ   Wear
8  Mess. Nr.   848
9  Gefä ß  Nr.  2
10 Datum   29.08.08
11 Zeit    09:04
12 Standreise  150
13 Alter   57
14 Kippwinkel(\dg) ␣300␣300␣318␣300␣300
15 [...]
```

Listing 3.1: Excerpt of a exemplary meta data file provided by HKM

Measurement data file. The measurement data file stores the measurement values of interest. The file is split into four columns separated by a tabulator. The first two columns reflect the two dimensional position inside a converter. In detail, column 1 holds the angle in degree and column 2 the depth in meter. Adding the radius stored in column 3 leads to a 3D position. The radius in meter is measured from the lot to the surface of the refractory material. Further on, the remaining brick value of the specified position is calculated and stored in column 4. Each row corresponds to one measurement value. Depending on the size of the converter up to 30000 measurement values exist. Listing 3.2 shows an excerpt of a typical laser measurement file produced by LaCam®.

```
 1  Angle   Depth   Radius  Brick
 2  0   0   2   --
 3  0   0.05    1.987   --
 4  0   0.1 1.973   93
 5  0   0.15    1.93    134
 6  0   0.2 1.924   148
 7  0   0.25    1.879   207
 8  0   0.3 1.851   255
 9  .   .   .   .
10  .   .   .   .
11  .   .   .   .
12  40  5.35    3.021   699
13  40  5.4 3.019   703
14  .   .   .   .
15  .   .   .   .
16  .   .   .   .
17  357.5   8.458   0.05    974
18  357.5   8.443   0   990
```

Listing 3.2: Excerpt of a exemplary measurement data file provided by HKM

Looking at row 2 in Listing 3.2, `'--'` indicates a missing value. Due to the measurement approach of LaCam® missing values occur frequently.

3.2.2. Production process data

Generally speaking, a heat is defined by approximately 100 production process parameters. HKM delivers one production process data file per campaign. This file stores a set of production process parameters for each heat. Table 3.1 basically describes the production process data file format where each row corresponds to one heat. Column `'KV-Alter'` corresponds to a consecutive heat number. HKM uses a so-called `'Schmelznummer'` denoted by the abbreviation *SNR* to uniquely identify each heat. *KV* standing for `'Konverter'` denotes the number of the blowing stands and `'Gefäß Nr.'` identifies the vessel which is used. Columns Parameter$_1$ to Parameter$_N$ contain the parameters based on which further analyses are made.

In order to be able to successfully perform analyses on input data sets and consistency has to be guaranteed. Since our input data set is not consistent at all, data pre-processing draws our attention. The following section explains the pre-processing steps investigated in APO.

KV-Alter	Datum	SNR	KV	Gefäß Nr.	Parameter$_1$	$\cdots$	Parameter$_N$
1	06/10/2008	741626	2	3	value$_{11}$	$\cdots$	value$_{1N}$
2	06/10/2008	641627	2	3	value$_{21}$	$\cdots$	value$_{2N}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
M	12/11/2008	542743	2	3	value$_{M1}$	$\cdots$	value$_{MN}$

Table 3.1.: Exemplary production process data file format provided by the cooperating steel-plant HKM.

3.3. Data pre-processing

Laser data as well as production process parameter can suffer from inconsistent data sets. Missing or falsified values can occur in both occasions. In order to perform sense-full analyses on the input data sets one has to eliminate those problems. This is done during the pre-processing phase covered in detail in the following.

3.3.1. Laser data

Since LaCam® is based on runtime measurement problems like

- areas could not be reached
- areas are reached more than once with different measuring results
- different step-sizes between the measurements

are implied. Additionally, measurement errors can occur which lead to outliers in the laser data set. Remedies applied in the APO approach will be introduced in the following.

As described in Section 3.2 each laser measurement provided by LaCam® comprises meta data and measurement data. Both can suffer from following problems

(1) File is missing.

(2) File is incomplete. e.g. missing coordinates in measurement file or invalid campaign number in meta data file.

In both cases the data couple is skipped during the import phase.

Consolidation to ensure uniqueness. The remaining laser data couples are passed through a consolidation step. Due to the runtime measurement method of LaCam® coordinates inside the converter may be measured multiple times with slightly different results. For further analyses each coordinate should be distinct with respect to its remaining refractory thickness. Hence, a consolidation step is necessary. This pre-processing step counts the number of replicates of each coordinate and forms the mean of the brick thickness for the replicate coordinate. All further brick thickness values of the replicated coordinates are eliminated.

Interpolation to guarantee equal step-sizes. As stated in Section 3.2.1 each coordinate is defined by depth and angle. Basically, the step-sizes between two depth values or two angles are constant:

- depth step-size $= 5\text{cm}$
- angle step-size $= 2.5°$.

Due to the runtime measurement method used by LaCam® the depth step-sizes may vary. Particularly in the bottom area of the converter much more measurements exist. For our analyses, fixed step-sizes are advantageous. Thus, a fixed grid with a step-size of 5cm is desired. For this reason, the laser data is linearly interpolated with respect to the depth step-size.

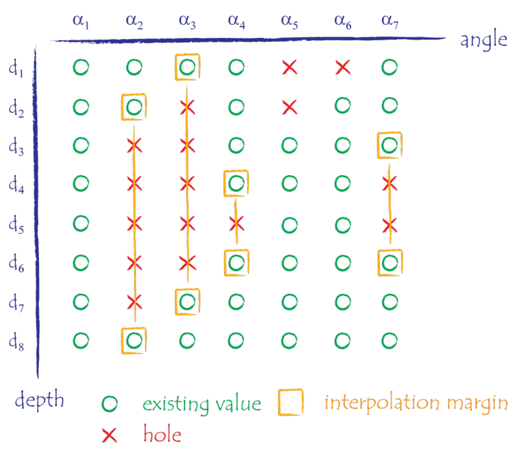
Eliminating holes. Some areas in the converter can not be reached by LaCam®, some coordinates do not have a brick thickness value. These aspects cause problems in the analysis step and have to be solved in advance. Therefore, a specific holes interpolation remedy is introduced. This is done in two consecutive steps

(1) interpolate depth

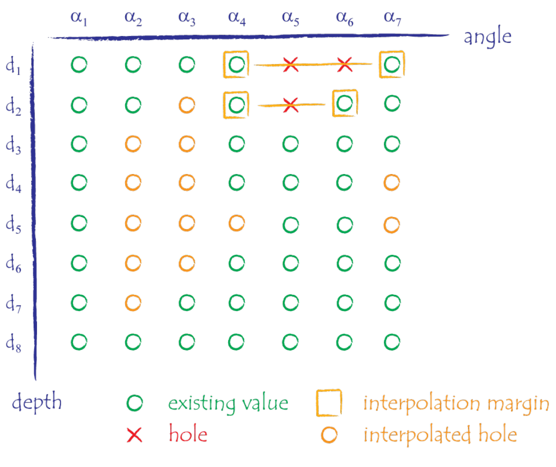
(2) interpolate angle.

In step (1) all joint holes within the same angle get linearly interpolated over all depths. This is done by finding the hole margins in both directions and linearly interpolating the missing values in between. After completing step (1), the remaining holes (mainly located at the converter margins) get linearly interpolated within the same depth over all angles. This is done in step (2) where again the same strategy is used. Looking at Figure 3.2 this strategy is summarized. Figure 3.2(a) shows step (1) performing the interpolation within each angle. The interpolated holes from step (1) as well as the remaining holes and their interpolation approach (2) are sketched in Figure 3.2(b).

Completing this holes interpolation the data set is optimally pre-processed and ready for further analyses. More detailed explanations and some additional implementation facts are added in Section 4.2.



(a) step (1): interpolate depth



(b) step (2): interpolate angle

Figure 3.2.: Two-step holes elimination strategy applied in APO. In the first step shown in (a) all joint holes within the same angle get linearly interpolated over all depths. In step two the remaining holes are linearly interpolated within the same depth over all angles as shown in (b).

3.3.2. Production process data

Since we have much more heats than LaCam® laser measurements we also have much more production process data than laser measurement data. As particularly described later in Section 3.4.1 we are only interested in one characteristic and representing value for each parameter between two laser measurements. Thus, missing values in the production process data set do not have a tremendous impact. For this reason, they would not be eliminated during the pre-processing phase rather than just skipped during the analysis phase.

The production process data pre-processing basically contains data structuring, date formatting and the calculation of the *availability* parameter introduced in the Section 3.4 particularly.

3.3.3. Merging Data

Once the laser data and the production process data are imported and pre-processed successfully they have to be combined in a sense-full way. Production process data is synchronized with laser measurement data based on

(1) heat number

(2) converter number

(3) vessel number.

Intuitively, SNR would be suggested as the best criteria to synchronize production process data and laser measurement data. Looking at the HKM data set more precise SNR is not a proper synchronization criteria. Directly after tapping the new SNR will be loaded at HKM. Thus, a LaCam® laser measurement will already result in the new SNR which is wrong. For this reason, SNR can not be used for synchronization purposes.

Summing up, in addition to the usual data import, pre-processing steps like consolidation, interpolation and holes elimination have to be performed to suffice the needs of the analyses steps in APO. Once the input data set is properly pre-processed, the desired analysis can be applied. Their APO specific customized approaches are covered in the following section.

3.4. Analysis

This section covers the analyses performed on the earlier introduced data set. First of all, some simplifications necessary in APO are described. Later on, the applied approaches and their specific usage as a part of APO are introduced.

3.4.1. APO approach

Since one of the main aims of APO is to find relationships between production process parameters and refractory wear, different approaches are investigated. In order to apply this approaches in a proper way the available data set has to be treated in a suitable manner. Thus, some simplifications have to be made which will be introduced in the following.

Production process parameter subset selection. Is it known from former research that finding relationships is not guaranteed. In fact, models with less parameters are easier and, in terms of computational complexity, cheaper to parametrize. For this reason, a top-down design is used to find rough relationships. Moreover, only 10 out of about 100 production process parameters are selected for the analyses in this thesis.

Based on the known negative influences on refractories illustrative depicted in the pyramid in Figure 3.3 the 10 most important available production process parameters are chosen. This is done in cooperation with refractory specialists from RHI AG. The chosen production process parameters as well as their nomenclature in production process data set, Figure 3.3 and APO software are summarized in Table 3.4.1. The parameter *availability* denoting the amount of heats per day is introduced and calculated by APO.

Analysis area definition. From Section 1.2 we know that converters are huge in terms of dimension and size. Thus, analysing the whole converter would be computational expensive. Furthermore, it is easier to parametrize smaller data sets. As a result, two

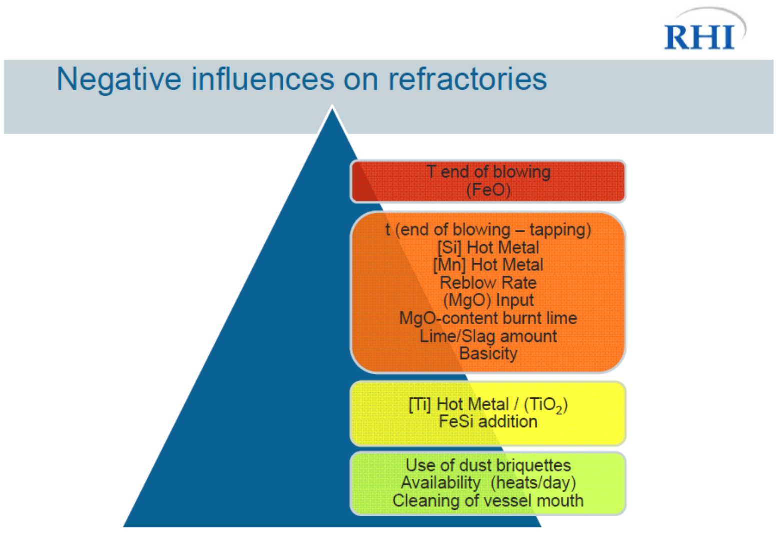


Figure 3.3.: Negative influences of production process parameters on refractory wear. Based on experience it is assumed production process parameters highlighted in red have the most influence while the influence amount decreases for orange, yellow and green boxes. (Source: [Jandl, 2009])

characteristic areas of the converter are selected to perform our researches. The area selection is done based on experience gathered by RHI AG specialists.

Basically, two areas are investigated

(1) area 8 - charge pad

(2) area 99 - virtual area without maintenance.

The charge pad area 8 is the most stressed converter area at HKM. Hence, the refractory wear is higher compared to other areas. Thus, maintenance is important in this area. In contrary to area 8, the virtually selected area 99 has much lower refractory wear and usually no maintenance is applied to this part of the converter. Since maintenance data is missing in this thesis, results on this area are assumed to be more accurate compared to results in area 8.

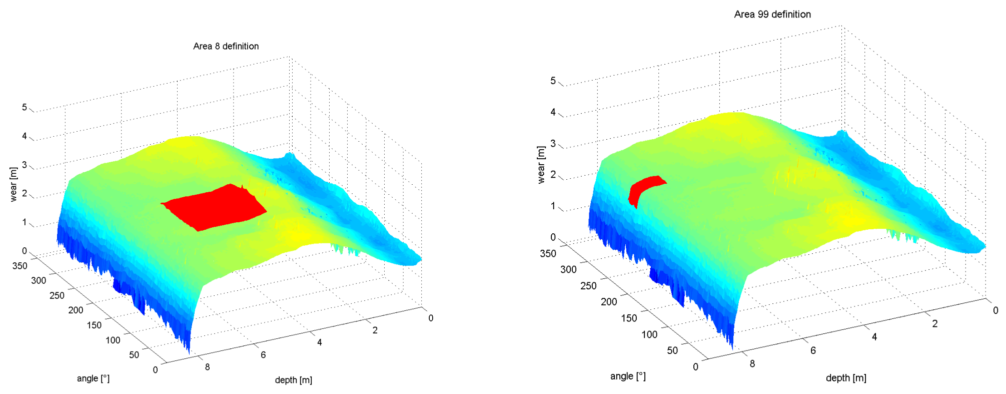
Area 8 is located between angles 130°- 230° and from 3.80m to 6.30m depth. Moreover, area 99 is slightly smaller and lies between angles 260°- 282.5° and 7m - 8m depth. Figure 3.4(a) visualizes area 8 and Figure 3.4(b) area 99 using red rectangles.

No matter how much data we have, it is essential to treat all available data equally. The APO approach to fulfil this requirement is described in the following.

Slot definition. The amount of production process data is much higher compared to laser measurement data representing the remaining refractory. As a matter of fact, production process data exist for each heat whereas in average only about 20 refractory

Nomenclature in Figure 3.3	production process data set	APO
T End of Blowing	TEMP ABSTICH	temp_abstich
(FeO)	Fe	Fe
t(end of blowing - tapping)	Liegezeit	liegezeit
[Si]	Si Re	Si_RE
[Mn]	MnRE	Mn_RE
Reblow Rate	Dauer nachgeblasen	dauer_nachgeblasen
(MgO) Input	MgO	MgO
Basicity	Basizitaet	basizitaet
FeSi	FeSi 75%	fesi_75prozent
Availability (heats/day)	-	availability

Table 3.2.: Selected most important production process parameters.



(a) area 8 (b) area 99

Figure 3.4.: Definition of analysed converter areas indicated by red rectangles. (a) highlights area 8 located in the charge pad area of the converter. The area 99 located close to the bottom of the converter is indicated in (b).

data measurements per campaign exist. In order to find relationships between production process parameters and refractory wear one is interested in the refractory wear between two laser measurements. In other words the refractory wear per heat in mm draws our attention. Moreover, the amount of heats applied between two laser measurements varies. Due to this facts, we defined so-called slots to equally treat all available data. A slot is defined as the part between two laser measurements, graphically visualized in Figure 3.5.

Knowing the amount of heats between two measurements and assuming linear refractory wear within a slot the refractory wear per heat in millimetre can be calculated like

$$rw_{heat} = \frac{\Delta rw}{slotsize} \tag{3.1}$$

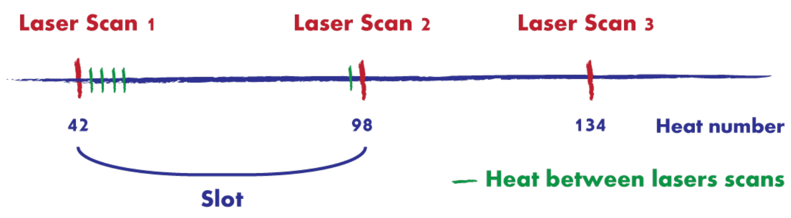


Figure 3.5.: Definition of a slot. A slot is defined as the amount of heats between two laser measurements. Since the refractory wear per heat draws our attention and slots varies in size it is required to normalize all measurements by the size of the corresponding slot.

where Δrw is the refractory material difference between one and its subsequent laser scan. This can be done either for each position or coordinate in the converter or for a characteristic value representing an area. The simplification of this characteristic values are covered in the next paragraph.

Characteristic functions define representing values. The approaches applied to find the desired relationships require parametrizations of model defining parameters. In fact, the resulting model should find proper relationships as accurate as possible on the one hand but without loss of generality on the other hand. For this reason and caused by the fact of increasing computational costs analyses are not performed for each coordinate inside a converter. As introduced earlier analyses are rather made for different areas inside a converter. As we will see in the subsequent sections all analysis methods apply a kind of regression approach. Hence, one production process parameter value acts as training and one Δrw value acts as target to perform the regression.

Having many refractory wear values per area and many production process parameters inside a slot lead to more than one couple of data. As a result, some simplifications have to be made. From the refractory point of view, the minimum value per laser scan of an area is used as representative characteristic value. The reason for this choice is easily argued. The minimum function represent the position with the lowest remaining refractory material. From a steel-plant point of view this is the most markable value in terms of maintenance and preservability. The characteristic function applied for production process parameters inside a slot varies from parameter to parameter. With the help of experienced refractory specialists from RHI AG the characteristic functions are selected. The main goal is to select the characteristic functions which has the most impact to the refractory wear. The selected characteristic function per production process parameter as well as their substantiation can be seen in Table 3.3.

Applying this simplifications, one ends up with one value per production process parameter per slot as training data and one refractory wear value per area per slot as target data. As a matter of fact, all requirements for successfully applying regression analyses are assured.

Production process parameter	Characteristic function	Substantiation
temp_abstich	max	$\uparrow$ tapping temperature $\rightarrow$ massive impact to wear of refractory material
Fe	max	(FeOn) makes the slag liquid and aggressive
liegezeit	max	$\uparrow$ liegezeit $\rightarrow$ higher refractory wear
Si_RE	max	$\uparrow$ (Si) $\rightarrow$ more heat is generated $\rightarrow$ higher refractory wear
Mn_RE	max	$\uparrow$ (Mn) $\rightarrow$ more heat is generated $\rightarrow$ higher refractory wear
dauer_nachgeblasen	max	risk of over-blowing $\rightarrow$ higher refractory wear
MgO	min	$\uparrow$ (MgO) $\rightarrow$ the stiffer the slag
basizitaet	min	$\downarrow$ basicity $\rightarrow$ more chemical reactions between slag and refractory material $\rightarrow$ higher refractory wear
fesi_75prozent	max	$\uparrow$ (fesi_75prozent) $\rightarrow$ energy added for additional heat $\rightarrow$ higher refractory wear
availability	min	$\downarrow$ converter temperature $\rightarrow$ higher refractory wear

Table 3.3.: Characteristic function definition per production process parameter and its substantiation.

In conclusion, some simplifications have to be made to ensure proper analyses. The definition of areas of interest, slots and characteristic values are covered in this section.

Supervised learning approach. Apart from the basic *LR*, all approaches applied in APO are based on the supervised learning principle introduced in Section 2.1.1. Recap, that each approach realisation consists of a training and a test phase. In terms of APO the available data set is split into 70% training set and 30% test set. The splitting is done completely randomly, independent of chronological issues.

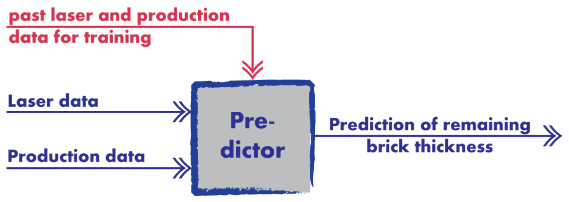
Data scaling before applying analysis approaches is very important especially for approaches like *SVR* and *GP*. It avoids parameters in greater numeric ranges dominating those in smaller ranges. Moreover, scaling is a proper remedy to overcome numerical difficulties during calculations. In APO each production process parameter is scaled to the range of $[-1, 1]$. It is important to apply the same scaling to both training and test data [Hsu et al., 2010].

Based on the training data set the model is parametrized. Having the model, predictions of the remaining data are made and compared to the real test data set. Figure 3.6 summarizes the data flow in 3.6(a) and schematically sketches the supervised learning principle applied in APO in 3.6(b).

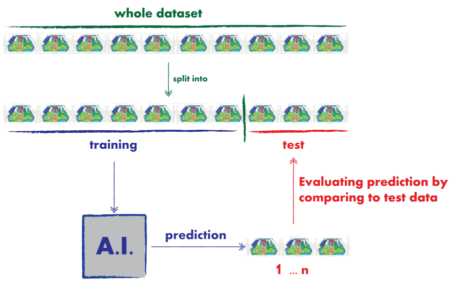
The following sections cover the used analyses more specific and explains the way of applying this methods in APO.

3.4.2. Linear Regression

The main principle of *LR* is explained in Section 2.1.3 in detail. Basically, in *LR* one tries to linearly fit the input data set. Thus, one has to parametrize the slope k and the offset d of a straight line defined by the equation of a straight line $\boldsymbol{y} = k\boldsymbol{x} + d$. In case of APO the 10 selected production process parameters act as input $\boldsymbol{x}$ while $\boldsymbol{y}$ is represented by the refractory wear. Thus, the parameters k and d can easily be determined by applying


past laser and production data for training
Laser data
Production data
Pre-dictor
Prediction of remaining brick thickness

(a) General APO data flow



(b) Schematic describing the data split.

Figure 3.6.: General data flow of APO and schematic data split into training and test set. All input data is passed into a predictor parametrized by past laser and production data in order to predict remaining brick thickness as shown in (a). The available data set is split into training and test set while the training set is used to train the artificial intelligence predictor ahead of comparing their prediction results to the real test set data. This typical supervised learning concept is depicted in (b).

least-squares fitting [Bishop, 2007]. First of all, only one production process parameter is considered for regression. Later on, production process parameters are combined and a multiple regression is performed.

3.4.3. Support Vector Regression

Section 2.1.4 introduced the theoretical background of *SVR* in general. Recap, the basic idea of *SVR* is to find a function $f(x)$ with at most ϵ deviation from observed targets for all training data on the one hand and is shaped as smooth as possible at the same time, on the other hand.

Different kernels used in APO. The mentioned Section 2.1.4 explained the meaning of kernels to make the analysis computational feasible to non-linear regression problems. Generally speaking, various kernels exist. In APO we investigated in two different kernels, namely

(1) RBF kernel

(2) polynomial kernel

The kernel selection is based on former researches like [Hsu et al., 2010] designating the RBF kernel as the kernel to start with, and the polynomial as a variant worthy to investigate in.

RBFs in general are a widely used regression model. It has the property that each basis function depends only on the radial distance from a centre or target value. This radial distance is typically designed as an Euclidean distance. Historically, exact function interpolations are the purpose of introducing these RBF functions [Bishop, 2007].

In APO the RBF kernel of the form

$$K(x_i, x_j) := exp(-\gamma||x_i - x_j||^2) \qquad \gamma > 0 \tag{3.2}$$

is used where γ is the kernel hyper-parameter.

The polynomial kernel formally denoted in Equation 2.13 is known in literature as a popular example for non-linear modelling and can be formulated for APO purposes like

$$K(x_i, x_j) := (\gamma x_i x_j)^d \qquad \gamma > 0 \tag{3.3}$$

where γ denotes again the kernel hyper-parameter and d the degree of the polynomial kernel.

***SVR* hyper-parameters.** As one can see, either polynomial or RBF kernels are parametrized by so-called hyper-parameters. From Equation 2.9 we know the penalty hyper-parameter C of the error term [Hsu et al., 2010]. The meaning of the kernel hyper-parameters γ and degree d in case of the polynomial kernel are mentioned earlier. In fact, finding the optimal hyper-parameters is the main goal of the *SVR* training phase to result in a model providing accurate prediction results.

Determine best hyper-parameters. Adapting from previous researches, for instance [Hsu et al., 2010], the hyper-parameters are applied in the binary numeral system, except the polynomial degree hyper-parameter, and their ranges are initially set to

- error deviation of loss function $\epsilon = 2^{-8} - 2^2$
- kernel function $\gamma = 2^{-16} - 2^3$
- penalty hyper-parameter $C = 2^{-6} - 2^8$

for *SVR* using RBF kernel and to

- error deviation of loss function $\epsilon = 2^{-8} - 2^2$
- kernel function $\gamma = 2^{-16} - 2^{-1}$
- penalty hyper-parameter $C = 2^{-6} - 2^6$
- polynomial degree $d = 2 - 4$

for *SVR* with polynomial kernel. Moreover, a grid search is applied to determine the best hyper-parameters. In doing so, step-sizes have to be defined. Initially the following step-sizes are used

- $\epsilon_{step} = 2^1$

- $\gamma_{step} = 2^1$
- $C_{step} = 2^1$

for both RBF and polynomial kernels and additionally $d_{step} = 1$ for the *SVR* with polynomial kernel.

Iterative grid search to speed up evaluation. During the hyper-parameter evaluation, it turns out, that this step becomes very time consuming and computational expensive. As a remedy an iterative grid search is investigated. Thus, the hyper-parameter selection is split into two consecutive parts. An initial rough grid search is applied using step-sizes like

- $\epsilon_{step} = 2^2$
- $\gamma_{step} = 2^3$
- $C_{step} = 2^3$.

Evaluate best hyper-parameters using cross validation (CV). In detail, the hyper-parameter selection is performed using a 20-fold CV . Generally speaking, CV is a famous method for limited training and test data sets. n-fold CV splits the whole data set into n equally sized proportions. Then, k trials are executed where all but one proportion is used for training purposes and the remaining part as test set. During a CV each proportions acts once as test set. Each trial results in an error. The average of all errors is represented by the mean squared error (MSE) defined in Equation 5.1. As a result, the hyper-parameter combination with the smallest MSE is selected.

Use finer grid search to improve accuracy. In second step of the iterative grid search, the area around the best hyper-parameters is analysed more detailed. Hence, the step-sizes are decreased to initially introduced values for all hyper-parameters and the hyper-parameter ranges are fitted to the margins of the area skipped by the rough grid search in the first step to ensure a finer grid search. Again, the best hyper-parameter combination is found by minimizing the MSE. Due to this iterative grid search approach a tremendous speed up of $50 - 70\%$ can be achieved.

Having the optimal hyper-parameters APO is able to train the model on the whole training set and performs predictions using the test data set.

3.4.4. Gaussian Processes

Gaussian Processes are initially introduced and explained in Section 2.1.5. To recap, a *GP* describes a probability distribution of functions where each function suffices a Gaussian distribution. A *GP* is completely specified by its mean $m(x)$ and covariance $cov(\boldsymbol{x}, \boldsymbol{x'})$ function, also known as kernel function in literature.

In fact, the modelling of either the mean function $m(x)$ or the kernel function $k(\boldsymbol{x}, \boldsymbol{x'})$ reflect the essential part of the *GP* training phase. Particularly, the procedure of applying *GP* for regression tasks is explained in Section 2.1.5.

Mean and covariance function selection. Referring to APO, the selection of mean and covariance functions and their initial hyper-parameter values are important in order to properly train the model. The prediction quality depends on the choice of the mean and covariance function. Based on former researches and statements in literature like [Bishop, 2007] the mean function $m(x)$ is set to zero. In contrary, the covariance function is not set to a fixed value rather than learned. In doing so, one first has to formally define the covariance function. In APO we use a combination of exponential, linear and constant kernel shown in Equation 3.4, which [Bishop, 2007] evaluated as proper covariance function for regression tasks. Looking at Equation 3.4, $\boldsymbol{\theta}$ represents the hyper-parameters and D the dimensionality of the input vector.

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \theta_0 \exp\left(-\frac{1}{2}\sum_{i=1}^{D}(x_{ni} - x_{mi})^2\right) + \theta_2 + \theta_3 \sum_{i=1}^{D} x_{ni}x_{mi} \tag{3.4}$$

During training phase the hyper-parameters $\boldsymbol{\theta}$ are optimized using log likelihood maximization mentioned earlier in Equation 2.27.

APO application of *GP ARD*. In Section 2.1.5 the *automatic relevance determination (ARD)* introduced by [Rasmussen and Williams, 2006] is mentioned. *GP ARD* offers the opportunity to easily represent the influence of each input production process parameter on the refractory wear by just adding a new hyper-parameter η. Knowing the exposure of the influence of a production process parameter on the refractory wear opens various new opportunities for APO and for their designated users. For instance, steel-plants are able to extract their top 3 wear production process parameters based on *GP ARD*. In doing so, Equation 3.4 is extended by η as shown in Equation 6.72 in [Bishop, 2007] leading to a kernel definition as mentioned in Equation 2.29.

3.5. Conclusion

In conclusion, this chapter covered the aim of this thesis with respect to APO, firstly. Further on, the input data set comprising production process data and laser measurement data is explained. Moreover, some necessary data pre-processing steps to guarantee proper analyses are explained. Finally, the applied concepts are covered with specific respect to APO. Since we know the routines that have to be passed through from a theoretical point of view, let us focus on the way of implementing all the procedures in the next chapter.

Teil IV.

IMPLEMENTATION

4 Implementation

Contents

This chapter basically focusses on the practical implementation. Section 4.1 generally introduces the framework applied in this thesis. As mentioned earlier, data pre-processing is essential to achieve proper analysis results. Thus, Section 4.2 introduces the APO data structure, explains the data import of production process data and laser measurements and covers the pre-processing steps necessary to ensure consistent data sets. Finally, Section 4.3 practically draws the attention to the implementation of the different analysis approaches and their prerequisites.

4.1. Framework

Generally speaking, MATLAB® is chosen as developing language for several reasons. In fact, MATLAB® is a high-level technical computing language useful for various tasks. Algorithm development, data visualisation, data analysis as well as numeric computations are suitable tasks. Flexibility and the availability of a huge amount of add-on is one of the main advantages with respect to APO. All these advantages are good reasons for choosing MATLAB® as the proper tool for the given problem. APO is based on several different analyses. Some are already integrated in MATLAB®, some are not.

In order to perform *SVR* or *GP* analyses additional libraries are necessary. Thus, the LIBSVM library and the GPML toolbox will be introduced in the following.

4.1.1. LIBSVM

The library provided by [Chang and Lin, 2011] was initially published in 2000. The department of Computer Science at the national Taiwan University in Taipei have been actively developing this package since the initial publishing. LIBSVM has been widely used in many different areas. Within 2000 and 2010 more than 250,000 downloads of the package have been recognized. Table 4.1 lists representative and successful works using LIBSVM in various domains.

Domain	Representative works
Computer vision	LIBPMK [Grauman and Darrell, 2005]
Natural language processing	Maltparser [Nivre et al., 2007]
Neuroimaging	PyMVPA [Hanke et al., 2009]
Bioinformatics	BDVal [Dorff et al., 2010]

Table 4.1.: Successful works with LIBSVM in various domains [Chang and Lin, 2011].

LIBSVM application. The typical use of LIBSVM involves two steps famous in supervised learning. Firstly, the data set is trained to obtain a model which is secondly used to perform predictions on a test data set. LIBSVM is implemented in C++ but provides interfaces for MATLAB®, Python etc. MATLAB® calls the LIBSVM functions and subroutines using so-called MATLAB® executables (MEX). Installation prerequisites for LIBSVM and MEX are shown in Section A.2 in Appendix A.

As a result, the *SVR* analysis with LIBSVM outputs the mean squared error (MSE) shown in Equation 5.1 as well as squared correlation coefficient R^2 shown in Equation 5.4 in the evaluation Section 5.1.

4.1.2. Gaussian Processes for machine learning toolbox.

From Section 2.1.5 we know, by definition, a *GP* is fully specified by a mean and a covariance function. From a practical point of view, these functions, typically given in terms of hyper-parameters, are mostly quite difficult to fully specify in advance. The difficulty is to infer the hyper-parameters from the data.

GPML toolbox overcomes hurdles. For this reason, the *GP* for machine learning (GPML) toolbox has been designed by [Rasmussen and Nickisch, 2010] to overcome these difficulties. They created a framework with the goal to be on the one hand, as simple as possible to use, and on the other hand, easy to extend. Moreover, the `GPML toolbox` is designed to show a high level of robustness while covering the full range of possible hyper-parameters. Stable and modular code checked by an exhausted suite of test cases is provided. The `GPML toolbox` is published under the FreeBSD license and offers, among others, full compatibility to `MATLAB`®.

GPML toolbox offers flexibility. One strength of the `GPML toolbox` lies in its flexibility. Mean, covariance and inference methods can be applied directly or combined to specialised functions. `GPML toolbox` calls the specialised functions composite functions. Additionally, likelihood functions are needed to properly apply a *GP*. In fact, likelihood functionality is needed both during the inference part of a *GP* and while predicting.

Based on the domain of interest the mean, covariance and likelihood functions as well as the inference method have to be selected. A proper combination of likelihood function and inference method depending on the task is essential to achieve well-performing results [Rasmussen and Nickisch, 2010]. A detailed technical documentation including all functions and methods is provided in the developer's guide on the `GPML toolbox` website[1]. Section A.2 in Appendix A states the `GPML toolbox` installation requirements.

4.2. Data pre-processing

As covered in Section 3.3 both laser measurement data and production process data are delivered in different file formats. Moreover, both can contain invalid or missing data. To avoid problems during the analysis each input data set has to pass a soundly pre-processing phase firstly. Once the raw input data is formatted in a suitable manner it can be imported and stored in a proper data structure. In this chapter the data structure used in APO is introduced in advance of describing the necessary pre-processing steps with respect to the implementation.

4.2.1. Data structure.

We know that a campaign is a collection of heats with each heat having production process parameters as well as laser measurements in the ideal case. In other words, production process parameters belong to heats and heats to campaigns which represents a kind of hierarchy. For this reason, `MATLAB`® `struct arrays` are used to store all input parameters. With a closer look, each campaign is uniquely identified using a `uid`. Moreover, campaign number, converter number, and vessel number are stored for each campaign.

[1]`http://www.gaussianprocess.org/gpml/code/matlab/doc/manual.pdf`

A boolean hasPData indicates whether this campaign has production process data or not. Additionally, each campaign holds a separate struct for each heat which stores few meta data tags like time t, data d, etc. as well as a boolean hasLData signalizing laser measurement availability. If laser measurements are available they are stored in the arrays angle, depth, radius and brick of the struct laserData. Finally, production process data is stored in the struct prodData with the name of the production process parameter as field name. Listing 4.1 summarizes the data structure used in this thesis.

```
c = struct( 'uid', 0, ...
            'nr', 0, ...
            'cv_nr', 0, ...
            'v_nr', 0, ...
            'hasPData', false, ...
            'heats',
            { ...
                struct( 'nr', 0, ...
                        'snr', 0, ...
                        't', '', ...
                        'd', '', ...
                        'hasLData', 0, ...
                        'laserData', ...
                        { ...
                            struct( 'angle', [], ...
                                    'depth', [], ...
                                    'radius', [], ...
                                    'brick', [],...
                                  )
                        },...
                        'prodData', ...
                        { ...
                            struct( 'param_1', 0, ...
                                    'param_2', 0, ...
                                    ...
                                    'param_N', 0, ...
                                 )
                        }
                       )
            }
           );
```

Listing 4.1: APO input data structure

Based on these data structure the raw data can be imported as described later in this chapter.

4.2.2. Production process data

HKM delivers production process data as a ∗.xls file, one file for each campaign. Since this file contains some statistics irrelevant for APO purposes the sheet of interest has to be extracted. This is done, by manually exporting it to a tabulator separated ∗.csv file to ensure a suitable format for the automatic import routine. In advance, it is essential to check whether the date column is delivered in the correct format. If not, the format has to be changed before exporting to the format `DD-MM-YYYY`. In terms of naming conventions, the exported production process data set is named fulfilling the pattern `ProductionData_<campaign_nr>_<converter_nr>_<vessel_nr>`.

Preparing the production process data in this way guarantees a successful import.

4.2.3. Laser Data

From Section 3.3.1 we know the sequential pre-processing order is defined as

(1) import data

(2) consolidation

(3) interpolation

(4) holes elimination.

Import data. In advance of pre-processing laser measurement data as described above, files have to be imported to our data structure. This is done in file `ImportData.m` by applying two steps. Firstly, laser measurement meta data is imported. Secondly, the actual laser measurement values are added.

As stated in Section 3.2.1 the meta data file contains a lot of unnecessary overhead. For this reason, regular expressions are used to fetch the data we are interested in. Listing 4.2 shows an excerpt of the function `fetchLMetaData` responsible for the meta data import.

```
SNR_NAME = 'Zustellung';
VESSEL_NAME = 'Gefä ß  Nr.';
CONVERTER_NAME = 'Stand';
DATE_NAME = 'Datum';
TIME_NAME = 'Zeit';
CAMPAIGNNR_NAME = 'Standreise';
HEATNR_NAME = 'Alter';

fid = fopen(f);

% Scan Text and separate each line
data = textscan(fid, '%s', 'Delimiter', '\n\r');

% Collect line of interest using regular expressions
expr = [DATE_NAME,  '\t\d+.\d+.\d+|', HEATNR_NAME, '\t\d+|', TIME_NAME, ...
                    '\t\d+.\d+|', CAMPAIGNNR_NAME, '\t\d+|', SNR_NAME, ...
                    '\t\d+|', CONVERTER_NAME, '\t\d+|', VESSEL_NAME, '\t\d+'];
[res] = regexp(data{1}, expr, 'match', 'once');
```

Listing 4.2: Import laser measurement meta data of interest using regular expressions

The real laser measurement values are far more structured than the meta data. Thus, the import is easily done by subroutine `fetchLData` invoking the `MATLAB`® function `textscan` using the tabulator as a column separator. Additionally, the laser measurement values are restructured to fit into our data structure by subroutine `reorganizeData`.

Finally, as mentioned in Section 3.3.1, missing and invalid parts are removed to complete the laser data import. Once the laser data set is imported and stored successfully different pre-processing steps covered in the very next paragraphs can be applied.

Consolidation. Referring back to Section 3.3 one recognizes the necessity of data consolidation to achieve unique data. Regarding the implementation, each heat of each campaign is passed through this consolidation executed in the file `ConsolidateData.m`. This file calls the subroutine `consolidate.m` performing some data extraction and formatting. The core consolidation is examined by a open-source `MATLAB`® function called `consolidator.m` provided by John D'Errico in 2009[2].

This consolidator can serve as a tool detecting replicates of each coordinate and results in one remaining representative value. Among others, John D'Errico's consolidator has the ability to determine the mean of the replicate values resulting in one remaining value and eliminating the replicates. This is important to enable a trouble-free use of `MATLAB`® interpolation method `interp1` necessary to guarantee fixed step sizes or interpolate measurement holes, in detail explained in the following paragraphs.

Interpolation. As described in Section 3.3.1 in detail, depth step sizes vary due the measurement approach of LaCam®. In order to achieve the desired fixed depth step size of 5cm the file `InterpolateData.m` has to be executed using the `strategy = 'depth'` switch. In doing so, function `interpolateDepth` is invoked and each laser measurement of each heat is linearly interpolated using the `MATLAB`® interpolation method `interp1`. Particularly, `interp1` finds the values of the underlying function $f(x)$ at intermediate data points and interpolates the found values [MATLAB, 2011]. In this thesis the interpolation is done linearly.

Eliminating holes. Section 3.3.1 introduced the necessity of holes elimination in detail and proposed a two-step interpolation strategy. Again, the file `InterpolateData.m` has to be executed, this time using the `strategy = 'holes'` switch to invoke the subroutine `interpolateHoles`. Initially, this function finds the hole margins using the subroutine `findHoleMargins`. Having these margins the holes in between is linearly interpolated using again the `MATLAB`® interpolation method `interp1`. As mentioned in Section 3.3.1 this is applied to all depths in advance to all angles. To ensure proper interpolation some boundary conditions have to be fulfilled. For further analyses all interpolated values as well as hole coordinates are saved to the data structure.

[2] `http://www.mathworks.com/matlabcentral/fileexchange/8354-consolidator`

Once the laser measurement holes are eliminated the laser measurement input data set is properly pre-processed for further analyses.

4.2.4. Merging data

To be able to perform the analyses we are interested in, laser measurement data and production process data have to be merged. This is done as described in Section 3.3.3 in function `mergeData`. It is remarkable that campaigns without production process data are tagged with `hasPData = 0` and heats without laser measurement information with `hasPData = 0` during this merging procedure.

Finally, the successful pre-processed and imported data set is saved using the APO data structure to a `MATLAB`$^{®}$ file called `data.mat`.

Summing up, input data sets are not consistent or flawless initially. Thus, pre-processing steps have to be considered. Moreover, the data has to be imported and stored using a suitable data structure. The data structure and all necessary steps to properly store input data in it are covered in this section referring to the implementation.

4.3. Analysis

As mentioned in Section 3.1 different regression approaches are applied to determine the best available relationships between production process parameters and refractory wear. The APO specific adaptations of this approaches are introduced in Section 3.4. This section focusses on the implementation of the simplifications applied in APO as well as the realisation of the approaches *LR*, *SVR* and *GP* using `MATLAB`$^{®}$ with the help of additional libraries like `LIBSVM` or `GPML toolbox`.

APO simpifications. Section 3.4.1 introduced the simplifications necessary for APO. Basically, all simplifications are applied in advance to the analysis application. Most simplifications are implemented in the file `ExtractRefractoryWearData.m`. Based on the output file `data.mat` of the importing APO part the desired data sets in terms of areas are extracted. Moreover, the production process parameters of interest applying their individual characteristic functions are extracted. Additionally, the slot sizes are calculated and the characteristic refractory wear values are determined. Particularly, this is done in a subroutine called `calcRWCharacteristics` returning the Δ_{rw} and the weighted rw_{heat} defined in Equation 3.1. Finally, `ExtractRefractoryWearData.m` calculates some area specific statistical values like `min(depth)` or `max(depth)`. The extracted data is saved to a file named sufficing the pattern `<#campaigns>c_area_<area_nr>_RefractoryWear.mat`.

4.3.1. Linear Regression

As stated in Section 3.4.2 the *LR* is implemented in two steps. Initially, only one production process parameter is considered for regression. Later on, productions process parameters are combined to apply *MLR*. Thus, followed by `ExtractRefractoryWearData.m` with `fieldOI='linRegData'` delivering the pre-processed data for *LR* purposes, two different files have to be executed. `AnalyseLinearRegression.m` performs the simple *Linear Regression* using one production process parameter while `AnalyseMultipleLinearRegression.m` has to be invoked if more production process parameters are combined.

***LR* using single production process parameter.** Basically, *LR* is performed for each heat having laser measurement data and each production process parameter separately. `AnalyseLinearRegression.m` fetches the necessary data and invokes a subroutine called `performLinReg` performing the actual *LR*. Listing 4.3 shows the subroutine `performLinReg`. In detail, the production process parameters `x` as predictors and refractory wear `y` as target are delivered. Line 15 uses the `MATLAB`® function `polyfit` to fit the polynomial, which is in this case the equation of a line, to the data. The resulting hyper-parameters are stored in `p` and used in line 18 to determine the prediction using the `MATLAB`® function `polyval`. Moreover, the residual values are calculated as shown in line 21. Lines 24 to 31 applies *LR* performance evaluation steps considered in Section 5.2.2 particularly.

```
1  function [p, rsq, yfit, yresid] = performLinReg(x, y)
2  % Performs linear regression
3  %function [p, rsq] = performLinReg(x, y)
4  % INPUT
5  %       x       - predictor values
6  %       y       - target values
7  % OUTPUT
8  %       p       - linear regression hyper parameter
9  %       rsq     - coefficient of determination
10 %       yfit    - predicted yfit
11 %       yresid  - residual values
12
13 % Calculate Linear regression
14 % p(1) = slope, p(2) = intercept of the linear predictor
15 p = polyfit(x, y, 1);
16
17 % Calculate predicted yfit
18 yfit = polyval(p, x);
19
20 % Compute residual values
21 yresid = y - yfit;
22
23 % Measurement of goodness by calculating coefficient of determination (R^2)
24 SSresid = sum(yresid.^2);
25 SStotal = (length(y)-1) * var(y);
26 rsq = 1 - (SSresid/SStotal);
27
28 if ~isnan(rsq)
```

```
    % round
    rsq = str2num(num2str(rsq,'%7.4f'));
end
```

Listing 4.3: Subroutine `performLinReg` performing the single production process parameter *LR* in APO

Once each heat and each production process parameter is analysed using *LR*, all results are collected, post-processed and saved to a MATLAB® file sufficing the pattern `<#campaigns>c_area_<area_nr>_linregresults.mat`.

***MLR* using combined production process parameters.** Similar to the *LR*, the *MLR* is performed for each heat equipped with laser measurement data. In contrast, production process parameters are combined to sets of different sizes. Sets from 2 to 9 combined parameters are applied. All possible production process parameter combinations are considered. Listing 4.4 sketches a pseudo code representing the procedure passed through in `AnalyseMultipleLinearRegression.m`. As a matter of fact, a different function is used to perform the *MLR* compared to the *LR*. *MLR* investigates in the MATLAB® function `regress` performing *MLR* using least-squares. Delivering the refractory wear `w_delta_rw` and the selected production process parameter combination `par` as shown in line 13 results to a set of outputs defined in lines 7 to 12 taken from the MATLAB® documentation. Lines 2 and 15 are used for performance evaluation purposes using CV. Section 3.4.3 mentioned this evaluation approach in detail. In the end, proper results are plotted by calling the subroutine `plotplotMultipleLinReg` as shown in line 18.

```
1   % Function for crossvalidation
2   regf=@(XTRAIN,ytrain,XTEST)(XTEST*regress(ytrain,XTRAIN));
3
4   for k = EACH_PARAMETER_COMBINATION
5
6       % perform multiple regression using regress
7       % B     = vector B of regression coefficients
8       % BINT  =   BINT of 95% confidence intervals for B
9       % R     =   vector R of residuals
10      % RINT  =   matrix RINT of intervals that can be used to diagnose outliers
11      % STATS =   containing the R-square, F statistic and p value
12              %           for the full model, and an estimate of the error variance
13      [B, BINT, R, RINT, STATS] = regress(w_delta_rw, par);
14      % Perform crossvalidation to calculate performance using MAE
15      cvMAE(k) = myCrossval('mse', par, w_delta_rw, 'predfun', regf, 'leaveout',1);
16
17      % Plot interesting results
18      plotMultipleLinReg(...)
19
20  end
```

Listing 4.4: Pseudo code functionally representing the APO implementation of a *MLR*

Finally, the best results are extracted and saved to a MATLAB® file sufficing the pattern `<#campaigns>c_area_<area_nr>_multilinregresults.mat`.

In conclusion, two different *LR* approaches are investigated, implemented in two different files using two different calculation approaches. The simple *LR* directly evaluates the regression in a subroutine whereas the *MLR* passes production process parameter combinations to the `MATLAB`® function `regress` to perform the multiple regression. In both occasions, data pre-processing is done in advance using `ExtractRefractoryWearData.m`.

4.3.2. *Support Vector Regression*

Section 2.1.4 introduced the general idea of non-linear *SVR* while Section 3.4.3 covered all APO related issues of this approach. This section focusses on the *SVR* specific data pre-processing and the necessary steps to properly implement *SVR* using the `LIBSVM` library in `MATLAB`®.

***SVR* pre-processing.** Principally, as for *LR*, `ExtractRefractoryWearData.m` is used to extract the data subsets to analyse and store them to `data.mat`. In fact, *SVR* analyses need more detailed data pre-processing. For this reason, `PrepareDataSVR.m` is implemented. As mentioned in the developer's guide[3] the library `LIBSVM` requires a specific data format. Thus, `PrepareDataSVR.m` invokes a subroutine called `convertDataToLIBSVMFormat`, one the one hand responsible for scaling the input data to the range of [-1,1] and on the other hand converting the input data to the desired `LIBSVM` format. In detail, all features are stored using a sparse matrix and written to a file sufficing the pattern `<#campaigns>c_area_<area_nr>_dataToAnalyse_scale.txt` using the function `libsvmwrite` provided by `LIBSVM`. Once the data is scaled and stored in the required format, the data set is randomly split into 70% training data and 30% test data using the subroutine `splitSVRData`. As a results, training data is saved to a file fulfilling the pattern `area_<area_nr>_dataToAnalyse_scale.tr` while test data is stored using the pattern `area_<area_nr>_dataToAnalyse_scale.te`. Once all this steps are negotiated, the data pre-processing is completed and everything is set for applying the analyses.

***SVR* execution sequence.** To put it bluntly, it is essential to execute pre-processing and analysis parts in the following sequential order to achieve proper analysis results:

(1) `ExtractRefractoryWearData.m`

(2) `PrepareDataSVR.m`

(3) `SVR_Analysis.m`

`LIBSVM` usage in APO. The actual analysis of interest is performed using `LIBSVM` by executing `SVR_Analysis.m`. Listing 4.5 shows a pseudo code functionally representing the main steps implemented in `SVR_Analysis.m`.

[3] http://www.gaussianprocess.org/gpml/code/matlab/doc/manual.pdf

```
 1  % Import data
 2  [input.train_lbl, input.train_inst] = libsvmread(area_<area_nr>_dataToAnalyse_scale.tr);
 3  [input.test_lbl, input.test_inst] = libsvmread(area_<area_nr>_dataToAnalyse_scale.te);
 4
 5  % SVR initializations
 6  param.svr_type = 3;     % 3 - epsilon-SVR
 7  param.kernel_type = 1;                      % 0 - linear, 1 - polynomial, 2 - radial basis function
 8  param.nfold = 20;       % n of n-fold Crossvalidation; 0 for leave-one-out CV
 9
10  for k = EACH_PARAMETER_COMBINATION
11
12      % fetch default parameter ranges
13      param.ranges = getDefaultParameterRanges(param.kernel_type);
14
15      % perform rough grid search
16      param = determineBestSVRParameter(input.train_lbl, tr_inst, param);
17
18      % set parameters for finer iterative grid search
19      param.ranges.step_c = 1;
20      param.ranges.step_gamma = 1;
21      param.ranges.step_eps = 1;
22
23      % perform finer grid search
24      param = determineBestSVRParameter(input.train_lbl, tr_inst, param);
25
26      % train the model
27      cmd = ['-s ', num2str(param.svr_type), ' -t ', num2str(param.kernel_type), ...
28              ' -c ', num2str(param.bestc), ' -g ', num2str(param.bestg), ...
29              ' -p ', num2str(param.besteps)];
30      if isequal(param.kernel_type,1)
31          cmd = [cmd, '-d ', num2str(param.bestdegree)];
32      end
33      results.model = svmtrain(input.train_lbl, tr_inst, cmd);
34
35      % perform predictions
36      [results.predicted_label, results.accuracy, results.decisionvalues] = ...
37                      svmpredict(input.test_lbl, te_inst, results.model);
38
39      % evaluate predictions
40      results.rmse = calcRMSE(input.test_lbl, results.predicted_label);
41
42  end
```

Listing 4.5: Pseudo code functionally representing the APO implementation of a *SVR* with LIBSVM

Initially, the desired data to analyse is imported using the LIBSVM function `libsvmread` as shown in lines 2 and 3. Lines 6 to 8 depict the initializations necessary to properly apply *SVR*. As one can see, all hyper-parameters are stored in a `struct param`. As introduced in Section 3.4.3 different kernels are applied and a 20-fold CV is used to find the best hyper-parameters. Comparable to the procedure in *MLR*, production process parameter subsets of size 1 to 10 are created and each possible combination is evaluated using *SVR* indicated through the loop in line 10.

Iterative grid search implementation. As described in Section 3.4.3 an iterative grid search is applied. Looking at the implementation, the default hyper-parameter ranges as well as the step sizes are initialised using the `struct param` in the subroutine `getDefaultParameterRanges` in line 13.

The first step of the iterative approach, executes a rough grid search by invoking the subroutine `determineBestSVRParameter` delivering the production process training parameter `tr_inst`, the refractory wear training data set `input.train_lbl` and the hyper-parameter `struct param` as shown in line 16. The subroutine `determineBestSVRParameter` steps through all possible hyper-parameter combinations. For each hyper-parameter combination the *SVR* model is trained using the `LIBSVM` command `svrtrain` including a 20-fold CV. Finally, the subroutine `determineBestSVRParameter` returns the optimal hyper-parameter combination found by minimizing the MSE.

The best hyper-parameter combination is stored in the `struct param`. In the second step of the iterative approach, the area around the best hyper-parameters is analysed more detailed. In fact, the grid search is becoming finer by decreasing the step-sizes. Lines 19 to 21 indicate the step-size decrease while the finer grid search is executed similarly as before by calling the subroutine `determineBestSVRParameter` with smaller step-sizes as shown in line 24.

Train model with best hyper-parameters and perform prediction. Completing the time consuming hyper-parameter selection, the *SVR* is all set to be trained on the whole training data set. Lines 27 to 32 of Listing 4.5 shows the APO specific parameter setting stored in the `string cmd`. Table 4.2 summarizes the delivered `LIBSVM` parameters and their meaning.

hyper-parameter	meaning
`-s svm_type`	set the type of the SVM 3 = epsilon-SVR
`-t kernel_type`	set type of kernel function (default 2) 0 – linear: u'*v 1 – polynomial: (gamma*u'*v + coef0)ˆdegree 2 – radial basis function: exp(-gamma*|u-v|ˆ2)
`-c cost`	set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
`-p epsilon`	set the epsilon in loss function of epsilon-SVR (default 0.1)
`-g gamma`	set gamma in kernel function (default 1/num_features)
`-d degree`	set degree in kernel function (default 3)

Table 4.2.: `LIBSVM` parameters applied in APO [Chang and Lin, 2011].

Finally, the *SVR* model is trained delivering the production process training parameters `tr_inst`, the refractory wear training data set `input.train_lbl` and the command `string cmd` as shown in line 33. The resulting model is used for predictions applied on the test set by invoking the `LIBSVM` function `svmpredict` delivering production process test parameters `te_inst`, the refractory wear test data set `input.test_lbl` and the model `results.model` as shown in line 36. The predicted results are stored in the `struct results` comprising the predicted refractory wear `results.predicted_label` as well as some other statistics. Finally, the results are evaluated by calculating the root mean squared error (RMSE) which will be covered in detail in the evaluation Section 5.2.3.

To sum up, additionally necessary pre-processing steps are required to properly apply *SVR* analyses. The beginning of this section focussed on this required steps. APO uses the library `LIBSVM` to implement *SVR*. This section gives some implementation-related remarks about their usage in APO as well as focussing on hyper-parameter selection strategies like iterative grid search using CV. Finally, it is important to guarantee the sequential execution sequence introduced in this section.

4.3.3. *Gaussian Processes*

The basic principle of *GP* is introduced in Section 2.1.5 and the specific APO approach of *GP* is covered in Section 3.4.4. This section covers the *GP* specific data pre-processing and highlights the main implementation strategies used to apply both *GP* and *GP ARD* with the help of the `GPML toolbox`.

***GP* specific data pre-processing.** Recap, that `ExtractRefractoryWearData.m` is used to extract the data of interest to a `MATLAB`® file called `data.m`. No matter, similar to *SVR*, some *GP* specific pre-processing steps have to be applied. This is done in `PrepareDataGP.m`. Particularly, this pre-processing procedure fetches the input data, scales it to a range of [-1,1] by simply invoking the subroutine `scaleData`, splits it to 70% training data and 30% test data using the subroutine `splitGPData` converts it in a proper matrix. Finally, `PrepareDataGP.m` saves the pre-processed data to a file named based on the pattern `<#campaigns>c_area_<area_nr>_70_preparedData_GP.mat`.

***GP* execution sequence.** Subsequent to the *GP* specific pre-processing the analyses of interest are performed by invoking the file `GP_Analysis.m`. To guarantee proper analysis results the following sequential execution sequence is essential:

(1) `ExtractRefractoryWearData.m`

(2) `PrepareDataGP.m`

(3) `GP_Analysis.m`

`GPML toolbox` usage in APO. The main `GPML toolbox` interface `gp.m` is invoked in the file `GP_Analysis.m`. In advance, APO specific function formulations have to be made. Listing 4.6 shows a pseudo code excerpt functionally sketching the *Gaussian Process* APO approach implemented in `GP_Analysis.m`.

```
1  % (1) GP definitions
2  param.meanfunc = @meanZero;
3  param.likfunc = @likGauss;
4  param.inffunc = @infExact;
5  if useARD
6      param.covfunc = {'covSum', {'covSEard','covConst','covLIN'}};
7  else
8      param.covfunc = {'covSum', {'covSEiso','covConst','covLIN'}};
9  end
10
11 for i = EACH_PARAMETER_COMBINATION
12
13     % (2) GP initial values
14     if useARD, L = ones(r,1); else L = 0; end
15     hyp(i).cov = log([L; 1; 1]);
16     param.sn = std(error);      % error of SVR results
17     hyp(i).lik = log(param.sn);
18
19     % (3) Optimise GP hyper-parameter
20     hyp(i) = minimize(hyp(i), @gp, -100, param.inffunc, param.meanfunc, ...
21                 param.covfunc, param.likfunc, inp.train_feat(:,ind), inp.train_tgt);
22
23     % (4) Saving ETA values if using ARD
24     if useARD hyp(i).eta = hyp(i).cov(1:r); end
25
26     % (5) Performing GP prediction
27     [output(i).m res(i).s2] = gp(hyp(i), param.inffunc, param.meanfunc, ...
28             param.covfunc, param.likfunc, inp.train_feat(:,ind), inp.train_tgt, ...
29             inp.test_feat(:,ind), inp.test_tgt);
30
31 END
```

Listing 4.6: Pseudo code functionally representing the APO implementation of a *Gaussian Process* using the `GPML toolbox`

In step (1) of Listing 4.6 the functional forms of the *GP* are defined and stored in `struct param`. Based on results of [Bishop, 2007] on former regression tasks using *GP* the mean function is set to zero in line 2. Since we assume Gaussian noise, line 3 selects the proper likelihood function. The inference function is set to exact inference in line 4 which suits perfect to our assumption of Gaussian likelihoods. Exact inference reduces the computation of mean and covariance of a multivariate Gaussian to a simple matrix algebra. The covariance function is selected based on Equation 3.4 in line 8. It has a composite form summing a linear term `covLIN`, a constant term `covConst` and an isotropic squared exponential term `covSEiso`. The switch `useARD` is used to distinguish between usual *GP* and *GP* including ARD. The *GP ARD* implementation is explained later in this chapter. The loop in line 11 signifies the application of the following steps to each combination of production process parameters.

After completing the *GP* definitions, it is essential to properly set the initial values of the hyper-parameters. This is done in lines 14 to 17. The initial values of the composite covariance function are set to $log(0, 1, 1)$. The variance of the assumed Gaussian noise is set to variance of the earlier computed *SVR* error = prediction - target. Again, the variance is delivered in logarithmic space.

Once the initial hyper-parameters are set, they are passed through the optimization function `minimize` in line 20 to optimise the hyper-parameters using an expectation propagation approximation to the marginal likelihood. Additionally, the `struct param` and the training features `inp.train_feat` representing the selected production process parameters and the training targets `inp.train_tgt` representing the training refractory wear are delivered. By setting the maximum amount of conjugate gradient (CG) iteration steps to -100, the optimization part is fully specified. Finally, the main interface `gp.m` is invoked to perform the optimization resulting in the optimal hyper-parameters stored in `struct hyp`.

Based on the optimized hyper-parameters the *GP* is executed in order to perform predictions on the unseen test data set. For this reason, in addition to the parameters delivered earlier the test features `inp.test_feat` and test targets `inp.test_tgt` are added and passed to the `GPML toolbox` function `gp.m`. As line 27 shows the predictive mean `output(i).m` and the predictive variance `res(i).s2` are returned.

APO implementation of *GP ARD*. As mentioned in Section 3.4.4 APO customers are extremely interested in *GP ARD*. Looking at the implementation, few slight changes to *GP* have to be applied. Referring back to line 8 in Listing 4.6 the definition of the covariance has to be replaced by the definition shown in line 6. `GPML toolbox` already provides a *GP ARD* covariance function named `covSEard` which is selected in line 6. Instead of using the isotropic squared exponential part a squared exponential covariance function with ARD is used. Additionally, the initial values in step (2) starting from line 14 have to be adapted. Depending on the amount of used production process parameters the size of the vector $\boldsymbol{L} = [1, 1, \ldots, 1]$ varies. However, the initial values of the covariance function are set to $log(\boldsymbol{L}, 1, 1)$. To extract the η values containing the exposure of the production process parameter a further step (4) has to be added. Line 24 selects the proper η values from the `struct hyp(i).cov` and stores it to `hyp(i).eta`. Apart from the recently mentioned differences all other steps are equal compared to the common *GP*.

Summing up, to successfully apply *GP* for APO purposes an additional pre-processing step has to be passed. It reshapes, scales and splits the data into its desired format. It is essential to follow the sequential execution described in this chapter. Further on, the `GPML toolbox` is used to perform the *GP* analyses. The definitions of mean, covariance and likelihood functions as well as inference methods suitable to APO are shown. Both common *GP* and *GP ARD* are supported by GPML. This section covered all mentioned issues from an implementation viewpoint.

4.4. Conclusion

In conclusion, this chapter briefly mentioned the framework and reasonable substantiations for the choice of languages and libraries with respect to APO. Further on, the design of the APO data structure is covered and import as well as pre-processing routines are explained. The last part of this chapter focussed on the implementation of the different analysis approaches. Some necessary simplifications are mentioned as well as the usage of `MATLAB`®, `LIBSVM` and `GPML toolbox` with respect to APO is covered to realise *LR*, *SVR*, *GP* and *GP ARD* regression analyses.

Up to now, all theoretical backgrounds, APO specific approaches as well as practical prerequisites and implementation details are covered. Thus, we are all set to perform the analyses of interest. The next chapter focusses on the evaluation and presentation of the analysis results.

Teil V.

EVALUATION

5 Evaluation

Contents

This chapter explains the result evaluations starting with the introduction of the applied evaluation metrics and methods in Section 5.1. Further on, the evaluation methods used to optimize the hyper-parameters of the different approaches are covered. Finally, the different applied approaches are compared to each other. REC curves are used to determine the influence of data set size and statistical hypothesis tests are investigated to evaluate which machine learning methods lead to the best prediction results. Section 5.2 covers these evaluation results.

5.1. Evaluation methods

In order to evaluate properly all APO approaches different metrics and methods are required. This section introduces the applied error metrics and explains the methods used to evaluate the analyses results.

5.1.1. Error metrics

Mean squared error (MSE). The mean squared error (MSE) is a simple and widely used error function. The sum of the squares of the errors are determined and averaged over the amount of data points N. In fact, the division by N allows to compare different data set sizes on an equal footing. An error is defined as difference between the prediction $f(\boldsymbol{x}_i)$ and the corresponding real target value t_i for each data point x_i. Moreover, the MSE is defined as an non-negative function which is zero if, and only if, the prediction exactly passes each training data point [Bishop, 2007]. Equation 5.1 denotes the MSE definition formally.

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (f(\boldsymbol{x}_i) - \boldsymbol{t}_i)^2 \tag{5.1}$$

Root mean squared error (RMSE). The root mean squared error (RMSE) is a very similar error metric. Additionally, the square root is applied to the MSE as shown in Equation 5.2. The RMSE ensures comparison ability of data sets with different data set sizes on the same scale and in the same units [Bishop, 2007]. As a matter of fact, the RMSE is especially useful when errors are both positive and negative. APO investigates in a subroutine called `calcRMSE` to calculate the RMSE.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (f(\boldsymbol{x}_i) - \boldsymbol{t}_i)^2} \tag{5.2}$$

Mean absolute error (MAE). The mean absolute error (MAE) is yet another prediction quality error metric. It is again a quantity to measure how close predictions are to the real observations. Particularly, the MAE is an average of the absolute errors between predictions and targets. Equation 5.3 gives the definition of the MAE. Using the subroutine `calcMAE` the MAE is determined in this thesis.

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} |f(\boldsymbol{x}_i) - \boldsymbol{t}_i| \tag{5.3}$$

Squared correlation coefficient (R^2). The squared correlation coefficient R^2, in literature also known as coefficient of determination, is generally used in the context of statistical models and provides a measure of how likely future prediction are. In other words it is a statistic offering information about the goodness of a model fitting. In general the squared correlation coefficient R^2 is defined as shown in Equation 5.4.

$$R^2 = \frac{\left(N\sum_{i=1}^{N} f(\boldsymbol{x}_i)\boldsymbol{t}_i - \sum_{i=1}^{N} f(\boldsymbol{x}_i)\sum_{i=1}^{N}\boldsymbol{t}_i\right)^2}{\left(N\sum_{i=1}^{N} f(\boldsymbol{x}_i)^2 - \left(\sum_{i=1}^{N} f(\boldsymbol{x}_i)\right)^2\right)\left(N\sum_{i=1}^{N}\boldsymbol{t}_i)^2 - \left(\sum_{i=1}^{N}\boldsymbol{t}_i\right)^2\right)} \tag{5.4}$$

In case of *LR* with only one independent variable the squared correlation coefficient R^2 is the square between the observed and the predicted values. Basically, the squared correlation coefficient R^2 ranges from 0 to 1. Applying the squared correlation coefficient R^2 to a *MLR* it accords the multiple correlation coefficients [Chang and Lin, 2011].

In terms of R^2 values interpretation, it is easy to say, that $R^2 = 1$ indicates a regression result perfectly fitting the data.

5.1.2. Regression Error Characteristic curves

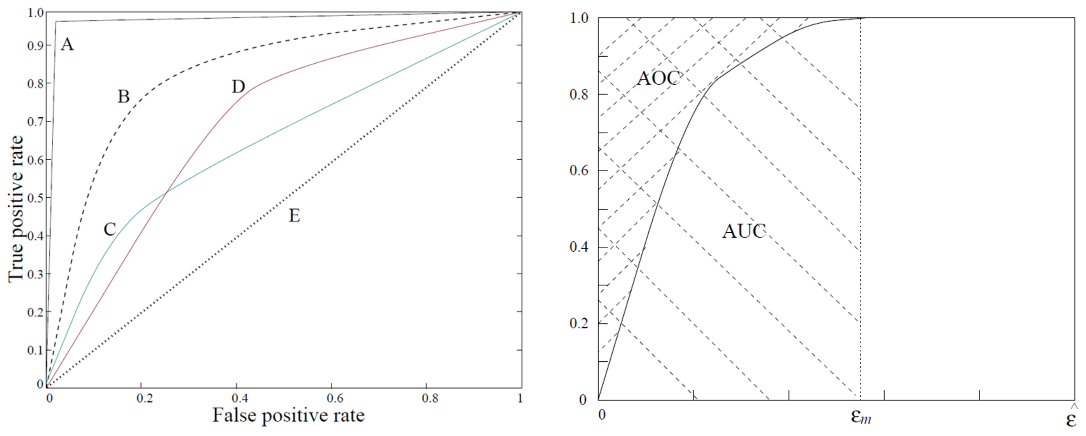
A powerful tool of performance evaluation are the *Regression Error Characteristic (REC) curves* introduced by [Bi and Bennett, 2003].

ROC curve principle. Generally speaking, *REC* curves generalize *Receiver Operating Characteristic (ROC)* curves, well known for visualizing and comparing classification results, to regression problems. Reviewing *ROC curves*, the fraction of true positives out of the positives (true positive rate) versus the fraction of false positives out of the negatives (false positive rate) is plotted. One can argue that a classifier performs well if the *ROC curve* climbs rapidly towards the upper left hand corner.

Looking at an *ROC* example in Figure 5.1(a) one can easily see that function A, which is an almost perfect model, dramatically outperforms the null model represented in function E. Summarizing *ROC* curves, a classifier is said to dominate another if the corresponding curve is always above the other.

The expected performance of a function can also be represented by the area under the curve (*AUC*) , which implies that the area over the curve (*AOC*) represents the expected error. Figure 5.1(b) sketches the definition of *AOC* and *AUC*. The *AOC* of an perfect model is 0. Note that *AOC* always is a biased estimate since it underestimates the actual expectation. This is due to dropping a part of the formula described in detail in [Bi and Bennett, 2003]. Depending on the used error metric, the *AOC* is either close to the *MAD* or to the *MSE*.

From ROC to REC. [Bi and Bennett, 2003] maintained all the properties and benefits of *ROC* curves and transformed them to regression tasks. Practically, *REC* curves plot the error tolerance versus the percentage of the points predicted within the tolerance. In other words, ϵ versus $acc(\epsilon)$ is plotted. In detail, the error tolerance is defined as the



(a) Sample ROC curves: function A = almost perfect model; E = null model.

(b) The area over the curve (AOC) estimates the expected error.

Figure 5.1.: Sample ROC curves in (a) and AOC definition in (b). A classifier performs well if the *ROC curve* climbs rapidly towards the upper left hand corner. From literature it is known that a classifier dominates another if the corresponding curve is always above the other [Bi and Bennett, 2003].

difference between the predicted value $f(x)$ and the actual value y. $acc(\epsilon)$ is defined like

$$acc(\epsilon) := \frac{|\{(\boldsymbol{x}, y) : loss(f(\boldsymbol{x}_i, y_i) \leq \epsilon, i = 1, \ldots, N\}|}{N} \tag{5.5}$$

where N denotes the amount of data points. It could be either the *squared residual*

$$loss(f(\boldsymbol{x}), y) = (y - f(\boldsymbol{x}))^2 \tag{5.6}$$

or the *absolute deviation*

$$loss(f(\boldsymbol{x}), y) = |y - f(x)|. \tag{5.7}$$

To plot REC curves the algorithm sketched in Listing A.1 in Appendix A implemented in the `MATLAB`® function `rec_curve` has to be invoked. The function plots the REC curve on the one hand and returns the calculated area over the curve (AOC) on the other hand. The developers [Bi and Bennett, 2003] describe the delivered arguments particularly.

REC advantages. In conclusion, the most important properties of an *REC* curve are

- REC curves provide visual comparison of regression functions with each other.
- The REC curve estimates the CDF of the error → area over the curve (AOC) is a biased estimate of the expected error.
- REC curve is invariant of error metric choice.

- REC curves provide an effective method to present results to non-experts.

5.1.3. Significance testing

Generally speaking, significance testing is a technique often used in statistics to encourage decisions. A decision is called statistically significant if a particular difference is caused by nothing more than coincidence. In other words, a decision is treated as significant if the likelihood that something happens is less than one in 20 [Rugg, 2007].

Significance testing was coined by [Fisher, 1925] like

> "Critical tests of this kind may be called tests of significance, and when such tests are available we may discover whether a second sample is or is not significantly different from the first." [Fisher, 1925]

Basically, a certain per-determined threshold probability exists to support a decision. This threshold is called significance level [Nickerson, 2000]. In doing so, an assumption, so-called hypothesis, is proofed using mathematical statistics based on empirical data sets. Initially, one usually starts with a null-hypothesis. By definition, the null-hypothesis assumes that there is no difference between two things. Further on, one tries to disprove the null-hypothesis by showing significant statistical differences between them [Rugg, 2007]. Practically, various different tests exist. Depending on the null-hypothesis and the domain of interest the suitable test can be selected.

APO uses Student's *t*-tests. Among others, the Student's t-test is often used to evaluate two groups. In fact, it is a statistical hypothesis t-test where the test statistic suffices a Student's t-distribution if the null-hypothesis is supported. As described in [Xue and Titterington, 2011] a Student's t-test is based on the assumption of having two groups following a normal distribution with non-equal means and equal but unknown within-group variances. One, basically, uses a null-hypothesis of assuming equal mean and tries to reject the null-hypothesis with a significance level of 5%.

Practical significance testing using `MATLAB`®. Regarding the implementation, the `MATLAB`® function `ttest2` performs the Student's t-test. The significance level α is set to 0.05 by default. Additionally, `ttest2` provides the ability to deliver an arbitrary α value. It returns a set of fields shown in Listing 5.1 in detail.

```
% Outputs:
%   - h         -   h = 1 => reject the null hypothesis
%                   h = 0 => failure of null hypothesis rejection
%   - p         -   the probability of observing the given result, or one more extreme,
%                   by chance if the null hypothesis is true.
%                   Small values of P cast doubt on the validity of% the null hypothesis.
%   - ci        -
%   - structure stats with the following fields:
```

```
%       'tstat' -   the value of the test statistic
%       'df'    -   the degrees of freedom of the test
%       'sd'    -   the pooled estimate of the population standard deviation
%                   (for the equal variance case) or a vector containing the
%                   unpooled estimates of the population standard deviations
%                   (for the unequal variance case)
```

Listing 5.1: Outputs of the `MATLAB`® function `ttest2` (Source: [MATLAB, 2011]).

In conclusion, based on the knowledge of the evaluation methods and metrics provided in this chapter, the APO analyses can be performed and rated. The next section reflects the achieved results.

5.2. Results

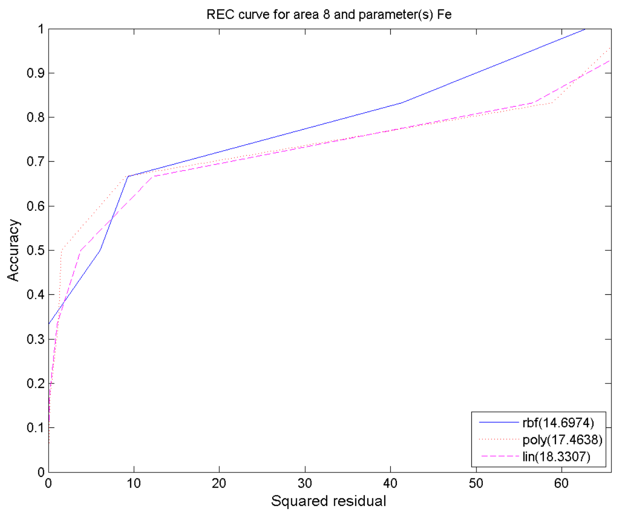
This section initially covers the influence of data set sizes. Finding the optimal hyper-parameters to design the model properly is important no matter which analysis approach is used. Thus, each approach is treated separately and the applied hyper-parameter evaluation strategies and their results are discussed. Finally, all approaches applied in this thesis are compared against each other to determine the most suitable approach for APO purposes. This final comparison is done in both, area 8 and area 99.

5.2.1. Importance of data set size

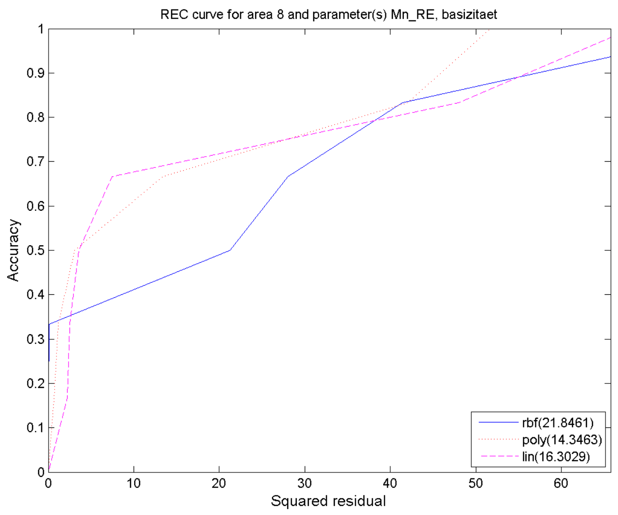
As mentioned in the previous sections, the data set is split into training and test set. In order to evaluate the influence of data set size, REC curves introduced in Section 5.1.2 are investigated. This thesis was started with only 1 available campaign to perform analyses. As time goes by, RHI AG and HKM provide more data sets.

For demonstration reasons, an arbitrary production process parameter combination is chosen and analysed in area 8. Looking at Figure 5.2 one can see that even for one production process parameter (e.g. Fe) or two combined parameters (e.g. Mn_Re, basicity) the results clearly improve by increasing the data amount. Figures 5.2(a) and 5.2(b) are analysed on just one campaign whereas Figures 5.2(c) and 5.2(d) uses 4 campaigns as their underlying data.

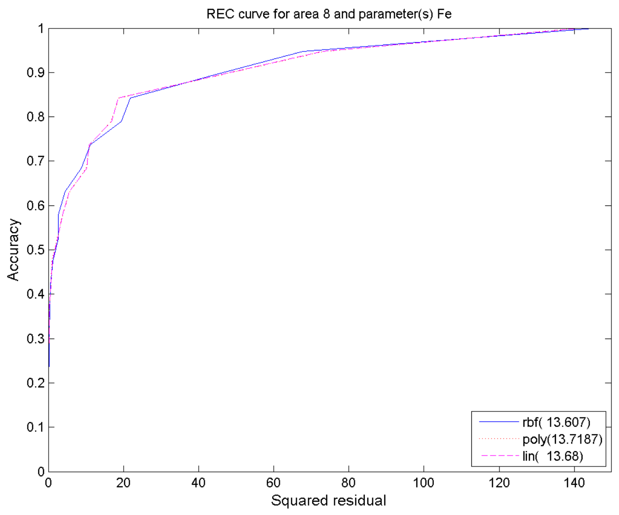
The more data the more accurate the predictions. From Section 5.1.2 it is known that the area over the curve (AOC) acts as a prediction quality measure. Moreover, a predictor performs well if the *REC curve* climbs rapidly towards the upper left hand corner. Since squared residual loss defined in Equation 5.6 is used, the AOC is close to the MSE. The value in braces in each legend represent the AOC. Comparing the results, one the one hand the AOC decreases by increasing the amount of campaigns, and on the other hand, the REC curves using 4 campaigns, visually climb closer toward the upper left hand corner.



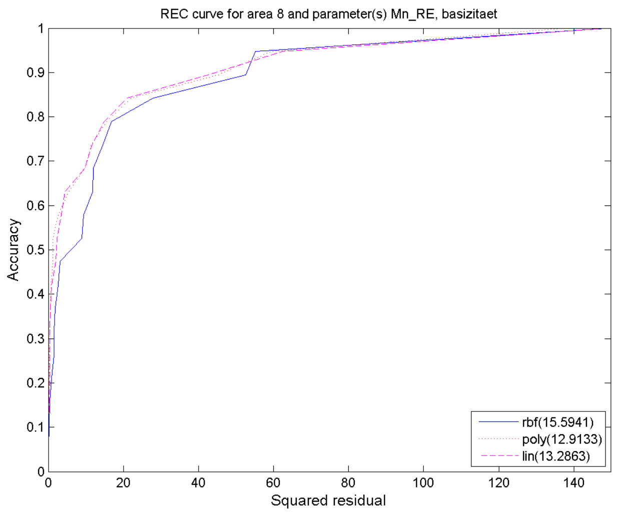
(a) 1 campaign, parameter Fe



(b) 1 campaign, parameters Mn_RE and basicity



(c) 4 campaigns, parameter Fe



(d) 4 campaigns, parameters Mn_RE and basicity

Figure 5.2.: Demonstration of data set size importance. An arbitrary production process parameter combination is chosen and analysed in area 8 using different data set sizes. (a) and (b) show the poor results achieved using just 1 campaign as input. By increasing the data set size to 4 campaigns in (c) and (d) clearly improved results for the same production process parameters can be found.

As a result, one can clearly see the tremendous influence of the data set size to the prediction results. By increasing the amount of campaigns the prediction results clearly improve.

5.2.2. *Linear Regression*

As already known, two different approaches of *LR* are investigated. The first treats each production process parameter separately and the second, combines production process parameters to evaluate synergy effects with respect to refractory wear. In both occasions the squared correlation coefficient R^2 introduced in Section 5.1.1 acts as a quality measurement. Recap, the better the linear model fits the data, the closer the R^2 value comes to 1. Referring back to Listing 4.3, lines 24 to 31 show the squared correlation coefficient R^2 calculation for the simple *LR* case. Applying *MLR* using the `MATLAB`® `regress` function, the squared correlation coefficient R^2 is returned in the `STATS` structure as shown in Listing 4.4.

Leave-one-out CV to evaluate results. Recollecting, *LR* and *MLR* models are the only approaches applied in this thesis not using the supervised training/test approach. This very first approach covers each campaign separately. The hyper-parameters defining the regression line, in the *LR* case, or the plane, in the *MLR* case, are directly observed during the regression. In order to evaluate the results a split into training and test set is still necessary. In case of *LR* and *MLR* this is done by applying a leave-one-out CV. In doing so, each data tuple acts once as test set while all others are used to determine the *LR* or *MLR*. Finally, the MAE defined in Equation 5.3 is calculated as an additional error metric.

Simple *Linear Regression* results. Focussing on each production process parameter separately does not allow to many conclusions on the relationships to the refractory wear. The squared correlation coefficients are smaller than 0.5 for each production process parameter. Most of them are even below 0.3. In average, over all 9 analysed campaigns and all production process parameters the R^2 lies below 0.1. Looking at the MAE values, the vary between 1.8 and 7.7 depending on the campaign and the parameter. In average the MAE lies around 3.3. Table 5.1 summarizes the *LR* results in area 8.

Summing up, using one production process parameter does not allow a detailed statement about the best production process parameter with respect to refractory wear. In other words, one is not able to state which production process parameter has the most influence on the refractory wear.

***Multiple Linear Regression* results.** By adapting *LR* to *MLR* the production process parameters are combined in advance of analysing their influences on refractory wear.

Campaign ID	MAE range	mean(MAE)	R^2 range	mean(R^2)
147_1_3	3.6883 - 4.0273	3.8546	0 - 0.2091	0.0447
148_2_3	1.8218 - 3.1818	2.6199	0 - 0.4857	0.1260
149_1_2	2.9118 - 3.1338	3.0113	0 - 0.0289	0.0124
149_1_3	4.9121 - 5.7668	5.5580	0 - 0.3308	0.0718
149_2_1	6.4733 - 7.7765	7.0635	0 - 0.2884	0.0826
150_1_1	2.6146 - 3.4096	2.9399	0.0067 - 0.2114	0.0700
150_2_2	3.1699 - 4.3512	3.6039	0 - 0.1788	0.0487
150_1_2	3.2781 - 4.1543	3.8094	0.0017 - 0.2523	0.0854
161_2_2	3.4342 - 3.7487	3.6493	0.0011 - 0.1497	0.0333

Table 5.1.: Results of simple *LR* in area 8 for each campaign.

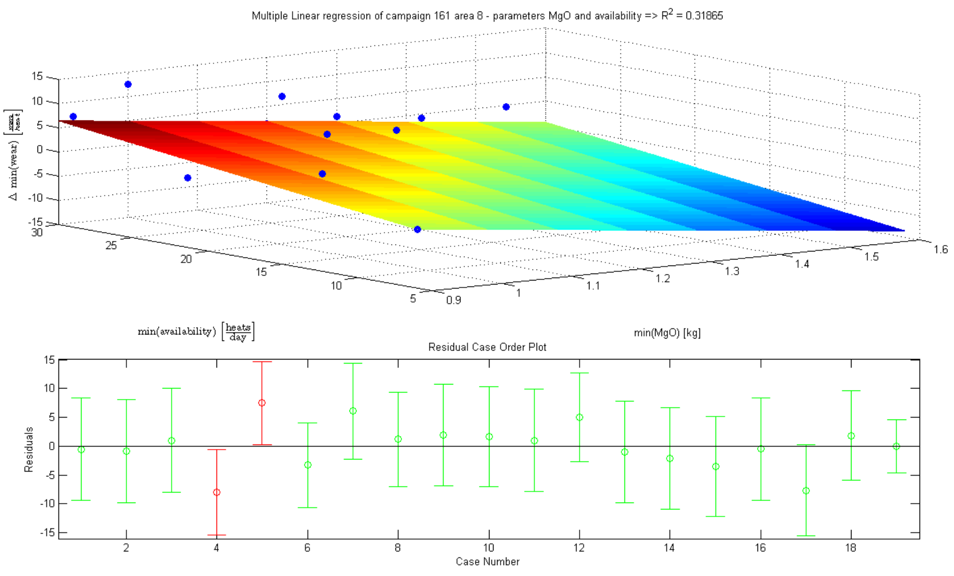
Particularly, combinations of size 2 to 9 are applied. Once, the combination size is specified, each possible combination of the 10 available production process parameters is evaluated. Similar to the *LR* a leave-one-out CV is used to determine the MAE of each production process parameters combination. Additionally, again the squared correlation coefficient R^2 is calculated.

Exemplary results for two combined production process parameters are shown in Figure 5.3 including the regression plane and the residuals. Interpreting the results in Figure 5.3(a), a higher availability leads to higher refractory wear whereas higher MgO decreases the refractory wear. With a closer look to Figure 5.3(b), one is able to argue, that higher 'liegezeit' and basicity has increasing influence on the refractory wear whereas lower 'liegezeit' and basicity, lead to the opposite effect. Looking at the title of both plots the squared correlation coefficient R^2 is much higher compared to the *LR* results indicating better fits.

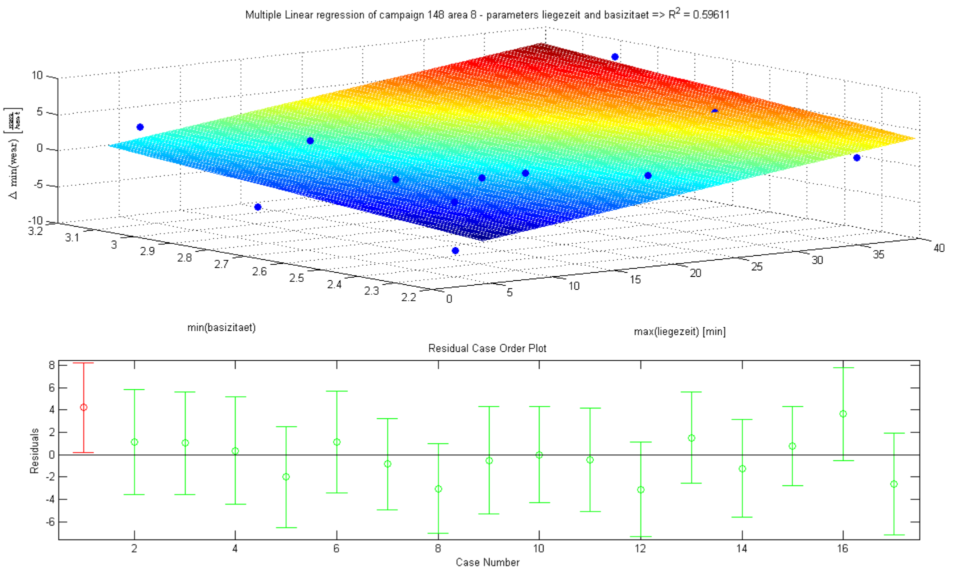
***MLR* results for more combined production process parameters.** Recap, parameter combinations greater than two are covered as well. Since they are difficult to plot in 3D, squared correlation coefficient R^2 values are analysed to evaluate *MLR* results. Table 5.2 shows the results for an exemplary campaign for all analysed combinations.

combination size	R^2 range	mean(R^2)
2	0.0047 - 0.3373	0.1602
3	0.0155 - 0.4017	0.2264
4	0.0443 - 0.4422	0.2856
5	0.0755 - 0.4986	0.3393
6	0.1157 - 0.5239	0.3886
7	0.2338 - 0.5390	0.4345
8	0.3499 - 0.5556	0.4780
9	0.4380 - 0.5202	0.5598

Table 5.2.: *MLR* results on campaign 151_1_2 for different combination sizes.



(a) Campaign 161_2_2, parameters availability and MgO.



(b) Campaign 148_2_3, parameters basizitaet and liegezeit.

Figure 5.3.: Exemplary *MLR* results including regression residuals achieved in area 8. The Δwear in mm/heat over two arbitrary combined production process parameters is plotted and fitted by a *MLR* plane. (a) shows that a higher availability leads to higher refractory wear whereas higher MgO decreases the refractory wear. The second example in (b) indicates that higher 'liegezeit' and basicity have increasing influence on the refractory wear whereas lower 'liegezeit' and basicity, lead to the opposite effect. Residuals are shown in the lower panels where red residuals represent the maximum or minimum values.

MLR does not perform equally well on each campaign. Some campaigns perform better than the exemplary campaign some lead to worse results than others. By interpreting the results one can see that they are dependent of the production process parameter selection. Some choices perform better than others. However, in conclusion, increasing the amount of combined production process parameters improve the *MLR* results.

To sum up, even the simple *MLR* model provides some insight to the relationship between production process parameters and refractory wear. But, as shown in this section, the relationship quality clearly depends on the selected production process parameters.

5.2.3. *Support Vector Regression*

Section 3.4.3 introduced two different kernels in which APO investigates. In contrast to the *LR*, the *SVR* does not analyse each campaign separately rather than using the whole data set and splitting it to training and test data set as described in Section 3.4. In doing so, the evaluation contains two steps. Firstly, the optimal hyper-parameters defining the *SVR* model have to be found using the training set. Secondly, the *SVR* model performance is evaluated on the test set.

Learn optimal hyper-parameters. As explained in Section 4.3.2 the optimal hyper-parameters are learned from the training data by applying a 20-fold CV minimizing the MSE_{train} on the training set. Similar to *MLR* all possible combinations of production process parameters with combination sizes ranging from 1 to 10 are covered. Using an iterative grid search the optimal hyper-parameter learning is accelerated. Increasing the combination size, the amount of possible production process parameter combinations increases up to a certain point. Thus, the hyper-parameter evaluation time grows just as much. Additionally, the results in terms of MSE_{train} does not improve for higher combination sizes. Table 5.3 summarizes the *SVR* hyper-parameter training results collected from analysing area 8 per combination size, including the learned hyper-parameters for the best production process parameter selection, the minimal MSE_{train} as well as the evaluation duration t_{eval} for *SVR* using both, RBF and polynomial kernel. For comparison reasons, Table 5.4 shows the same results for area 99.

Looking at the results in Table 5.3, there are slight differences in terms of hyper-parameter quality reflected in MSE_{train} between RBF and polynomial kernels. Moreover, the MSE_{train} decreases by increasing combination size up to 5 combined production process parameters for polynomial kernels whereas no improvements are made using RBF kernels. As already mentioned earlier the evaluation time t_{eval} grows up to a certain combination size. In case of area 8 it grows until combination size 5. The evaluation time is smaller for evaluating RBF than polynomial kernels caused by the additionally evaluated hyper-parameter degree d in case of the polynomial kernel. Focussing on

combination	RBF kernel					polynomial kernel					
size	ϵ	γ	C	MSE_{train}	t_{eval} [sec]	ϵ	γ	C	d	MSE_{train}	t_{eval} [sec]
1	1	0.5	64	24.4179	63.54	2	0.5	64	4	33.8585	67.97
2	1	0.5	64	24.4179	339.51	4	0.5	64	4	31.994	309.22
3	1	0.5	64	24.4179	990.59	1	0.5	64	2	30.4422	923.60
4	1	0.5	64	24.4179	1904.99	1	0.5	64	4	25.0739	1830.39
5	1	0.5	64	24.4179	2451.68	1	0.5	64	3	24.4179	2460.12
6	1	0.5	64	24.4179	2110.23	1	0.5	64	3	24.4179	2454.80
7	1	0.5	64	24.4179	1207.99	1	0.5	64	3	24.4179	1735.35
8	1	0.5	64	24.4179	459.16	1	0.5	64	3	24.4179	813.44
9	1	0.5	64	24.4179	102.41	1	0.5	64	3	24.4179	226.41
10	1	0.5	64	24.4179	10.39	1	0.5	64	3	24.4179	27.47

Table 5.3.: Learned optimal hyper-parameters for area 8 using *SVR* with RBF and polynomial kernels for different combination sizes.

hyper-parameter learning results of area 99 summarized in Table 5.4, basically, likewise results can be found. In case of area 99, both, RBF and polynomial kernels, slightly decrease their performance results indicated through MSE_{train} up to a certain combination size. Looking at the evaluation time t_{eval} the learning process has the longest duration for combination size 5 in terms of RBF kernels and 6 in terms of polynomial kernels.

combination	RBF kernel					polynomial kernel					
size	ϵ	γ	C	MSE_{train}	t_{eval} [sec]	ϵ	γ	C	d	MSE_{train}	t_{eval} [sec]
1	2^{-9}	0.5	64	6.1416	75.60	2^{-5}	0.5	64	2	7.1956	66.25
2	2^{-9}	0.5	64	6.1416	483.96	1	0.5	64	3	6.9629	313.94
3	2^{-9}	0.5	64	6.1416	1768.09	0.25	0.5	64	4	6.4998	956.83
4	2^{-11}	1	128	5.5214	4272.27	0.5	0.5	64	4	6.2831	2116.79
5	2^{-11}	1	128	5.5214	5277.96	2^{-9}	0.5	64	3	6.1416	3935.13
6	2^{-11}	1	128	5.5214	4103.42	2^{-9}	0.5	64	3	6.1416	5098.34
7	2^{-11}	1	128	5.5214	2169.46	2^{-9}	0.5	64	3	6.1416	4052.62
8	2^{-11}	1	128	5.5214	249.26	2^{-9}	0.5	64	3	6.1416	1953.44
9	2^{-11}	1	128	5.5214	155.27	2^{-9}	0.5	64	3	6.1416	506.74
10	2^{-11}	1	128	5.5214	14.80	2^{-9}	0.5	64	3	6.1416	56.50

Table 5.4.: Learned optimal hyper-parameters for area 99 using *SVR* with RBF and polynomial kernels for different combination sizes.

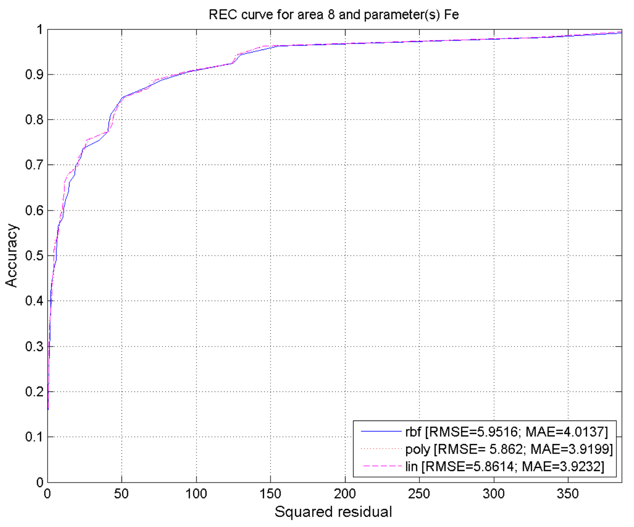
By comparing hyper-parameter results, area 99 clearly outperforms area 8 in terms of MSE_{train}.

Based on the hyper-parameters for each combination size the *SVR* can be finally trained and evaluated on the test set. In the following, the evaluation results are covered.

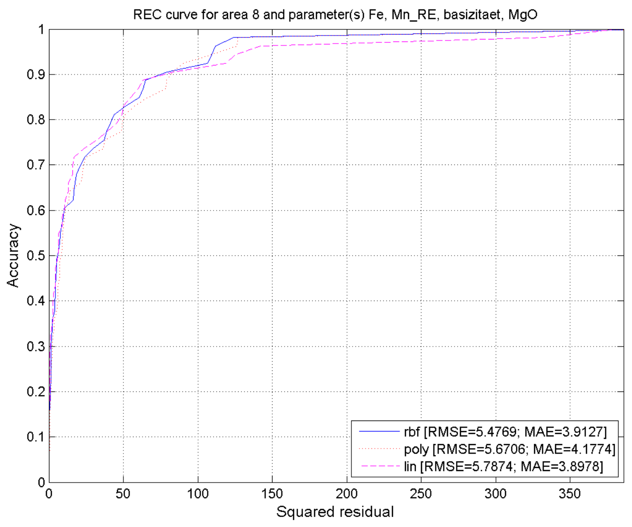
***SVR* results.** The *SVR* model is evaluated on the test set as described in Section 4.3.2. The REC curves introduced in Section 5.1.2 are used to visualize and compare the results. Additionally, the error metrics RMSE and MAE are investigated. Figure 5.4 shows the *SVR* results using RBF kernels, polynomial kernels and the *MLR* results for area 8. Looking at the results, one can see that using one production process parameter no real performance differences between the three compared methods can be found. By increasing the combined production process parameters to 4 as shown in Figure 5.4(b) the performance slightly increases indicated by the decreasing RMSE. Again, one can not argue that one approach outperforms another. Moving, to 6 combined production process parameters as shown in Figure 5.4(c) an interesting fact arises. Comparing the RMSE values to the earlier results, higher errors occur. As a result, combining to many production process parameters forces the prediction performance to decrease again. Moreover, the performance of *SVR* using polynomial kernel is getting worse compared to the other approaches. Finally, Figure 5.4(d) combines all 10 available production process parameters. The RMSE values are getting even more worse. Additionally, real performance differences between the approaches can be found. Looking at the REC curve shapes, one can see that the *MLR* performs slightly better than the *SVR* with RBF kernel. Both outperform the *SVR* with polynomial kernel.

Figure 5.5 summarizes the *SVR* results using RBF kernels, polynomial kernels and the *MLR* results for area 99 using the same production process parameters as in Figure 5.4. Similar behaviours can be observed comparing the results to area 8. One major difference exists. The prediction performance, in general, no matter which approach is used is much higher in area 99 compared to area 8. Thus, the RMSE values are much lower. The reason for this prediction performance improvement is easy to find. Area 8 is one of the most stressed areas of a converter at HKM. Thus, maintenance is important in area 8, whereas in area 99 usually no maintenance is required. Since maintenance data is not available so far, the machine learning approaches applied in this thesis are not able to predict area 8 as good as area 99. Referring back to the results in area 99, applying 1 production process parameter shown in Figure 5.5(a), as well as 4 combined production process parameters shown in Figure 5.5(a), does not lead to real differences in prediction quality between the two *SVR* approaches. Similar to area 8, more combined production process parameters badly influence the prediction performance. Thus, the RMSE values for 6 combined production process parameters shown in Figure 5.5(c), start to increase again. The *SVR* approaches do not perform as good as the *MLR* approach. With a closer look to the results using 10 combined production process parameters in Figure 5.5(d) increasing RMSE values and a *MLR* result outperforming both *SVR* kernels can be observed. In addition, the *SVR* kernel with polynomial kernel performs better than the RBF kernel.

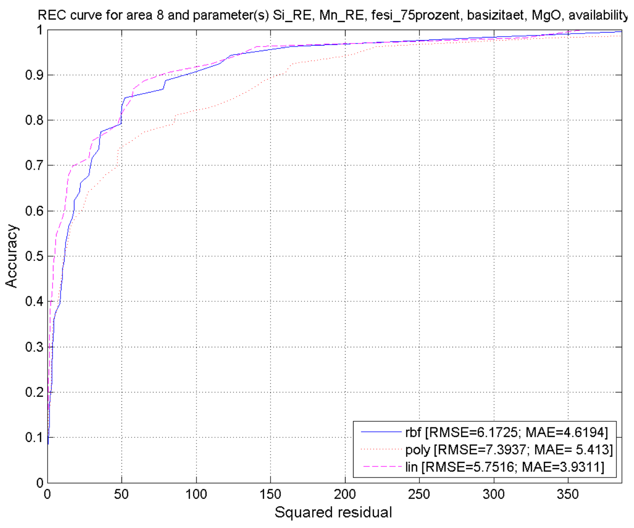
Summing up, *SVR* approaches are defined by a kernel which is modelled by different hyper-parameters. During a time-consuming iterative grid search the hyper-parameters are optimised. Based on the hyper-parameters the *SVR* is trained on the training set and evaluated on the test set. In conclusion, results for combined production process parameters improve up to a certain combination size and are getting worse again for too many combined production process parameters. Moreover, *MLR* outperforms the *SVR* especially for bigger combination sizes. It is shown in this section that results in area 99 area more accurate compared to area 8.



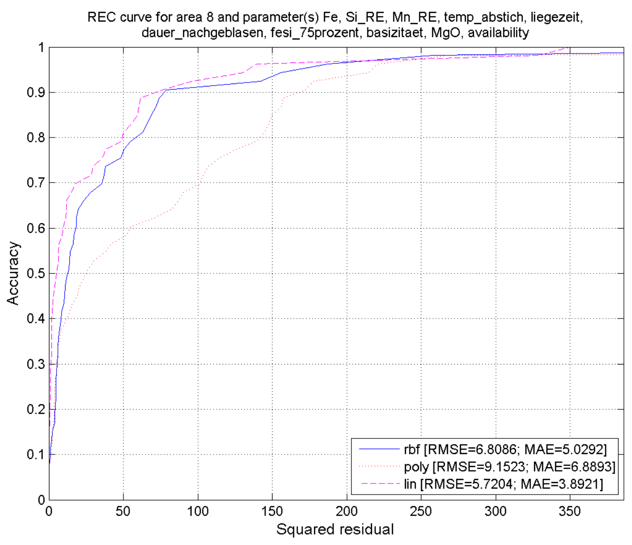
(a) REC plot using 1 parameter in area 8



(b) REC plot using 4 parameter in area 8

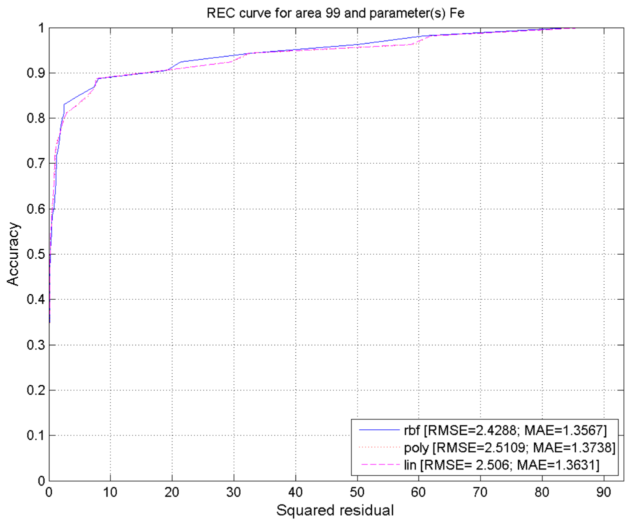


(c) REC plot using 6 parameters in area 8

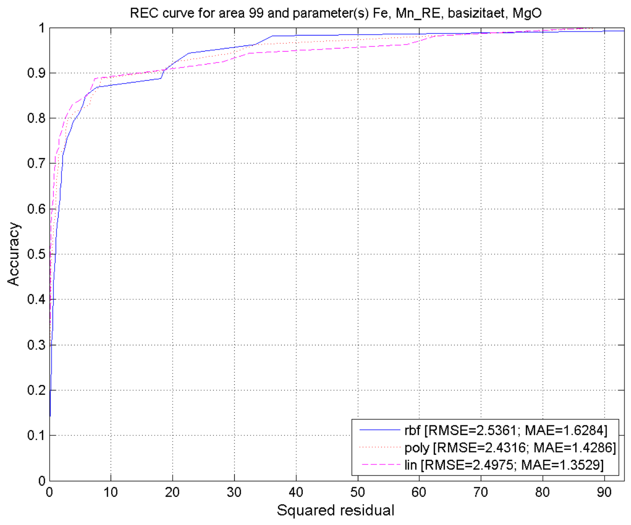


(d) REC plot using 10 parameter in area 8

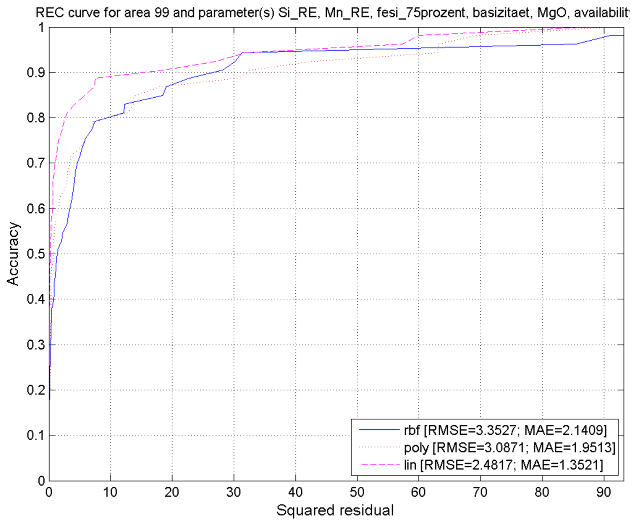
Figure 5.4.: *SVR* results using RBF and polynomial kernels compared with *MLR* results in area 8. No prediction performance improvements can be found for one parameter in (a). 4 combined production process parameters slightly improves the results of all approaches as indicated by the decreasing RMSE in (b). Increasing the combination size to 6 as shown in (c) the prediction accuracy decreases again. Same tendency can be observed by looking at combination size 10 in (d). The prediction accuracy again decreases and the linear approach performs better than the *SVR* with RBF kernel while both outperform the *SVR* with polynomial kernel.



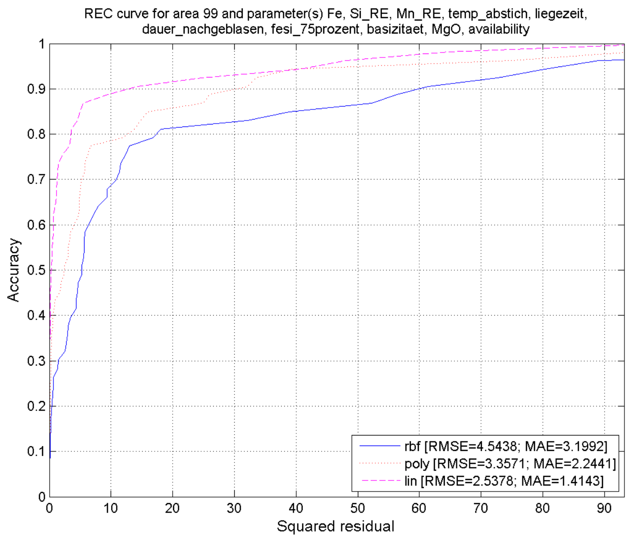
(a) REC plot using 1 parameter in area 99



(b) REC plot using 4 parameter in area 99



(c) REC plot using 6 parameters in area 99



(d) REC plot using 10 parameter in area 99

Figure 5.5.: *SVR* results using RBF and polynomial kernels compared with *MLR* results in area 99. Apart of the clearly improved prediction quality, similar behaviours as in area 8 can be observed. Analysing 1 production process parameter in (a) as well as 4 combined production process parameters in (b) does not lead to real differences in prediction quality between the different *SVR* approaches. By looking at RMSE values for 6 combined production process parameters in (c) one can see that they start to increase again, indicating worse results for higher combination sizes. (d) shows that for combination size 10 the linear model clearly performs better than both *SVR* kernels while the polynomial kernel slights performs better than the RBF kernel.

5.2.4. *Gaussian Processes*

As introduced in Section 3.4.4 two different methods of *GP* are applied in APO: basic *GP* and *GP ARD*. Similar to *SVR* approaches the data set is split into training set, from which hyper-parameters are trained to define the *GP* model, and test set used to evaluate the model.

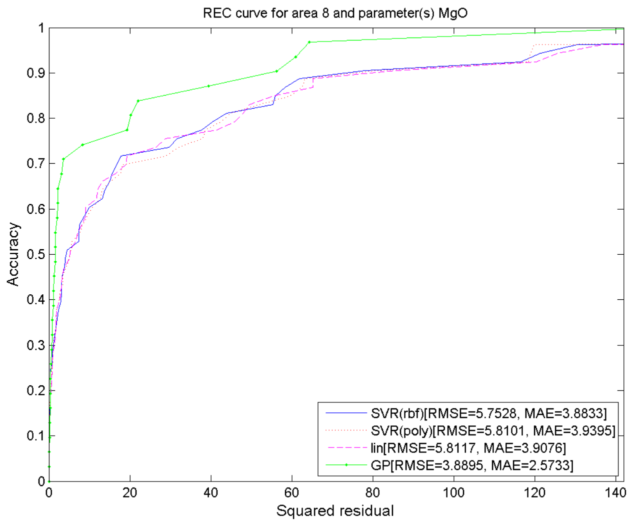
Learn optimal hyper-parameters. Particularly, Section 4.3.2 introduced the selection of covariance functions and the hyper-parameter training strategy based on initially set hyper-parameter values. Similar to *SVR* approaches, all possible combinations of production process parameters with combination sizes from 1 to 10 are covered. Again, hyper-parameter training time increases until combination size 5 and decreases for bigger combination sizes. Table 5.5 summarizes these training durations for area 8 and area 99.

	time [sec]			
combination	*GP*		*GP ARD*	
size	area 8	area 99	area 8	area 99
1	10.42	5.70	7.18	7.75
2	45.25	33.85	46.59	38.72
3	118.82	101.58	95.95	107.40
4	234.60	173.54	192.88	209.88
5	297.57	288.29	222.62	330.95
6	246.48	201.66	313.65	251.87
7	165.00	125.50	158.55	168.30
8	48.38	54.75	62.50	96.13
9	11.60	13.82	14.53	22.53
10	1.23	1.28	1.71	3.39

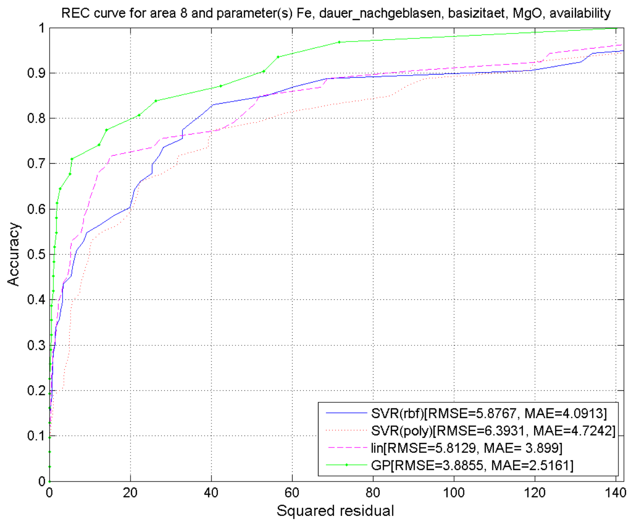
Table 5.5.: Duration of the *GP* hyper-parameter optimization for different combination sizes in area 8 and 99.

***GP* results.** The test procedure of *GP* is already mentioned in Section 4.3.3. Similar to *SVR*, REC curves and the RMSE as well as the MAE are investigated. Looking at the results in Figure 5.6(a), one can see that *GP* clearly outperforms the other approaches even for 1 production process parameter. The RMSE values indicate smaller errors for *GP* and quite equal errors for the other methods. Increasing the combination size to 5 combined production process parameters does not change the result magnificently. Again, *GP* leads to pretty much the same performance. The linear model improves

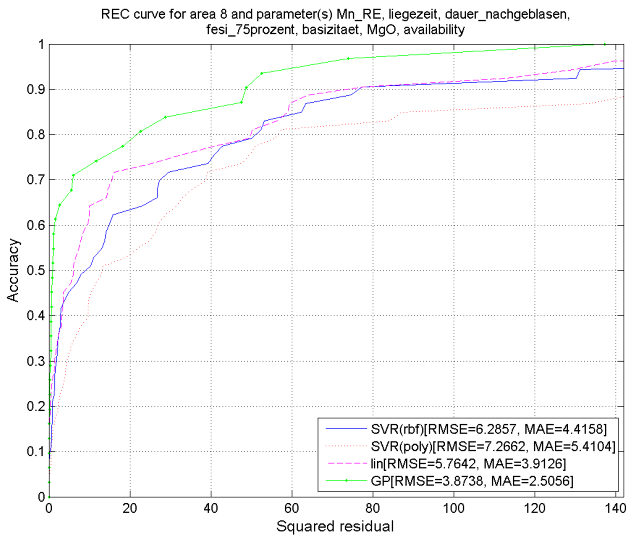
and leads to a slightly better performance than the *SVR* kernels. Figure 5.6(b) summarizes the results in detail. This tendency is approved when looking at Figure 5.6(c), where 7 production process parameters are combined. *GP* performs best, linear model improves and outperforms both *SVR* kernels. *SVR* with polynomial kernels leads to the



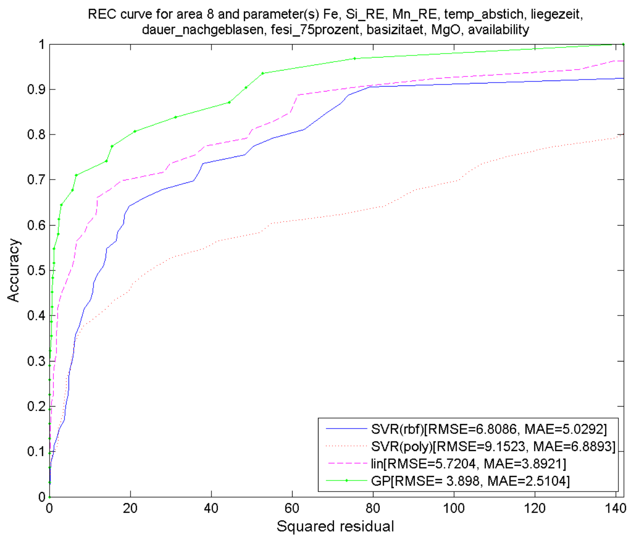
(a) REC plot using 1 parameter in area 8



(b) REC plot using 5 parameter in area 8

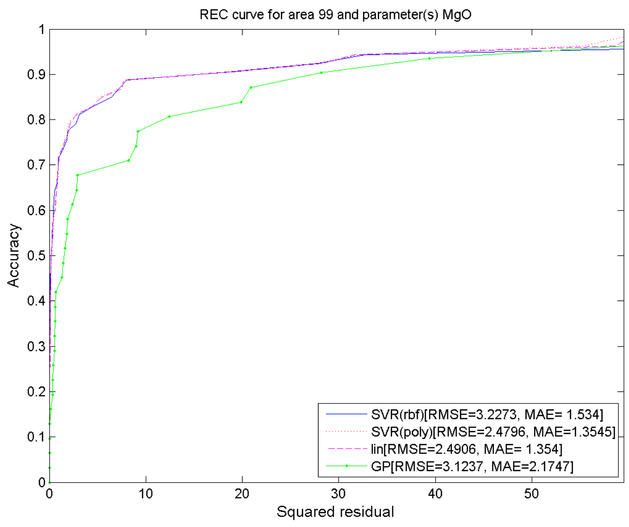


(c) REC plot using 7 parameters in area 8

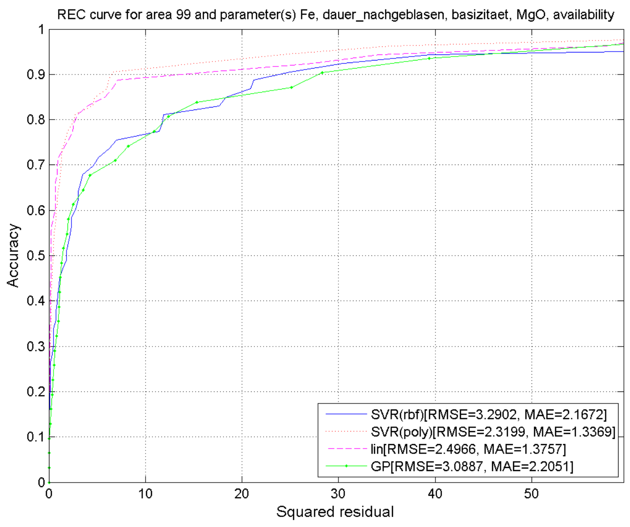


(d) REC plot using 10 parameter in area 8

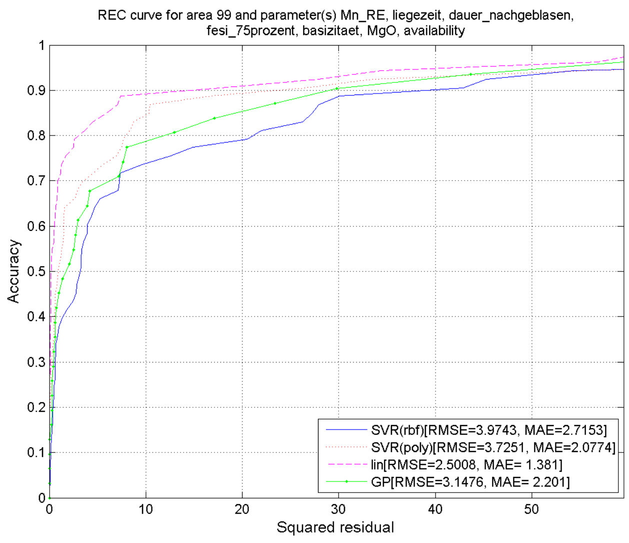
Figure 5.6.: *GP* results compared with other approaches in area 8. (a) shows that *GP* clearly outperforms the other approaches even for 1 production process parameter. Increasing the combination size to 5 or 7 combined production process parameters as shown in (b) and (c) does not change the results magnificently. *GP* still has the best prediction performance while the *MLR* improves and slightly performs better than the *SVR* kernels. (d) reflects a clear prediction performance ranking. The *GP* is superior to all other approaches, while the *MLR* results clearly perform better than both *SVR* kernels. Again, *SVR* with polynomial kernel leads to the worst results.



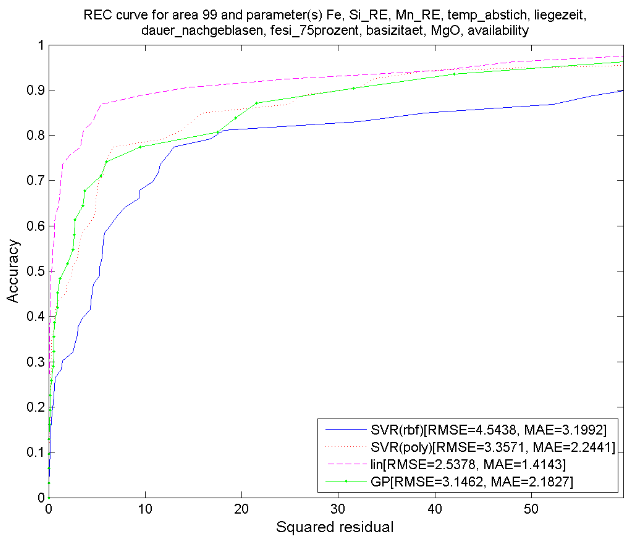
(a) REC plot using 1 parameter in area 99



(b) REC plot using 5 parameter in area 99



(c) REC plot using 7 parameters in area 99



(d) REC plot using 10 parameter in area 99

Figure 5.7.: *GP* results compared with other approaches in area 99. In contrary to area 8, *GP* even performs worse than conventional approaches for 1 parameter in (a). The prediction performance of *GP* is improved by increasing the combination size. It catches the *SVR* results for combination size 4 in (b) and even performs better than *SVR* with respect to the RMSE for combination size 7 in (c). In both cases the *MLR* concept leads to the best performance results. Combining 10 production process parameters in (d) approving this tendency. *MLR* results are still superior followed by the *GP* while *SVR* predictions lead again to the worst results.

worst result. Finally, looking at the combination of all 10 production process parameters, a clear ranking in terms of performance can be observed. *GP* leads to the best prediction results, ahead of the linear model, which again outperforms the non-linear *SVR* models where RBF kernels result better than polynomial kernels. Summing up, *GP* performs pretty well in area 8 and always leads to the best performance results.

Figure 5.7 shows the results of the same productions process parameter combinations in area 99. In the beginning, only 1 production process parameter is considered. In contrary to area 8, *GP* does not perform better than the other approaches. Its RMSE value is much higher and the area over the curve as well. By combining 4 production process parameters as shown in Figure 5.7(b), the prediction performance of *GP* is improving. It catches the *SVR* with RBF kernel in terms performance. *SVR* with polynomial kernel and the linear model still outperform the other two approaches. Figure 5.7(c) reflects the prediction performance achieved by combining 7 production process parameters. Both *SVR* kernels decline in terms of performance. Comparing the RMSE, the *GP* performs slightly better than the *SVR* methods. The linear model clearly outperforms the other concepts. Finally, Figure 5.7(d) summarizes the prediction performance results using all 10 production process parameters as a combination. The earlier observed tendency continues. The linear model leads to the best results. *SVR* kernels decrease more and more. *GP* performance remains quite stable and leaves both *SVR* kernels behind them.

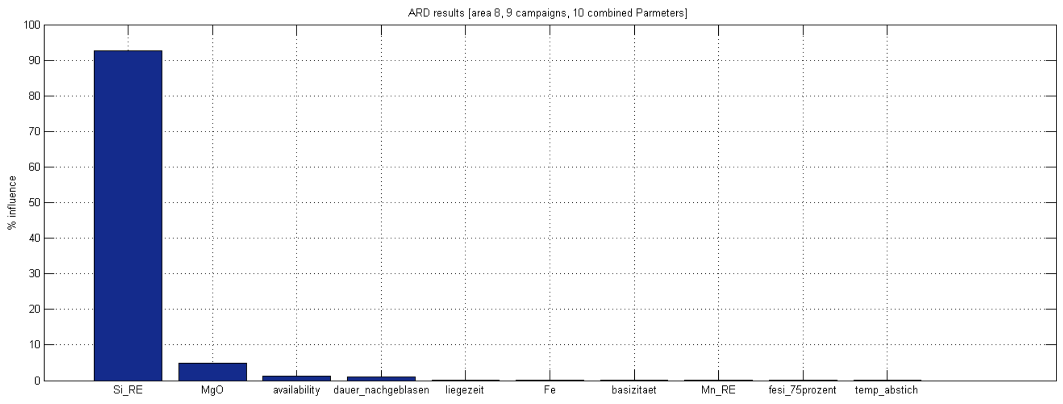
Summing up, *GP* does not perform that well in area 99 than in area 8 but in general results are better in area 99 indicated in smaller RMSE values. By increasing the combination sizes, the *SVR* performances decrease while linear and *GP* remain stable. Thus, linear models perform best in area 99 followed by *GP*.

5.2.5. *Gaussian Processes* with *ARD*

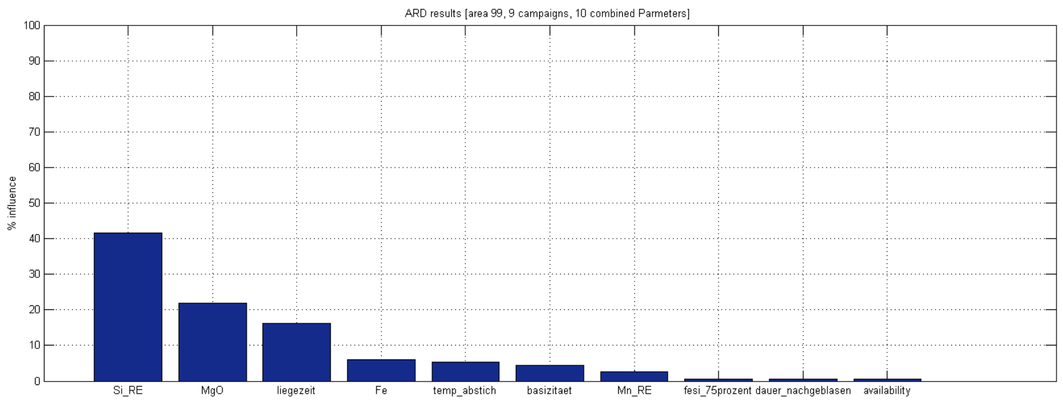
Basically, *GP ARD* results are equal to the *GP* results. The same hyper-parameters are found leading to the same prediction performances. Additionally, the η hyper-parameter introduced in Section 3.4.4 is determined. η allows deeper insight to the influence of each production process parameter on the refractory wear. Due to this additional hyper-parameter the training duration increases slightly compared to the basic *GP* approach. Columns 4 and 5 in Table 5.5 summarize the *GP ARD* training duration.

***GP ARD* results.** Figure 5.8(a) shows the influences of the 10 production process parameters on the refractory wear in area 8 in percent. One can see, that production process parameter *Si_RE* has the biggest impact. All other production process parameters have less than 10% influence on the refractory wear.

Similarly, Figure 5.8(b) reflects the influences found in area 99 in percent. Here, the influence factors are much more balanced where again *Si_RE*, *MgO* and *'Liegezeit'* hold the major parts.



(a) Production process paramter influence on refractory wear in area 8.



(b) Production process paramter influence on refractory wear in area 99.

Figure 5.8.: Weighted influences of production process parameters on refractory wear determined by *GP ARD*. Looking at area 8 in (a) production process parameter *Si_RE* plays a dominating role in terms of refractory wear influence. Influence in area 99 in (b) is much more balanced. Production process parameters *Si_RE*, *MgO* and *'Liegezeit'* have the biggest refractory wear influences.

***GP ARD* results interpretation.** Refractory wear specialists from RHI AG evaluated the results. The *GP ARD* results found for area 99 properly fit the experience those specialists have. Looking at area 8, RHI AG specialists only partly agree with the results found by APO. Due to missing maintenance and refractory material information as input, RHI AG is still quite satisfied with the results gathered by APO. As only 10 out of 100 production process parameters are chosen, not considered production process parameters can have even bigger influences.

In conclusion, *GP* approaches perform a conjugate gradient based optimization approach in order to determine the optimal hyper-parameters. Using this hyper-parameters the *GP* is trained on the training set and evaluated on the test set. Concluding the results, one is not able to say which approach leads to the best performance, in general.

Certainly, it depends on the analysed area and on the amount of combined production process parameters. By tendency, the linear model and the *GP* perform better than the non-linear *SVR* results. In order to evidently propose the best approach statistical tests are investigated. Their results are covered in the following section.

5.2.6. Performance evaluation

Since our data set is still very small and results vary depending on the selected production process parameter combination, one is not clearly able to argue which method has the best overall performance. Thus, statistical tests are investigated to find significant differences. Fur our purpose Student's *t*-test introduced in Section 5.1.3 are applied.

APO remedy to small test set. A small data set implies a small test set. In order to achieve qualitative results by applying the Student's *t*-test the small test set problem is overcome using the following approach. First of all, the hyper-parameter are determined as up to now. Then, two different test sets are created. One the one hand, the small real-test-set is used, and on the other hand, the whole available data set is used as test set again. From a practical point of view, the latter is not admissible since the same data is used to train the hyper-parameters. To qualitatively evaluate the significant differences between the APO approaches this is a irresistible treatment. In the following, a leave-one-out CV on the test set is performed for each approach. As a result, the RMSE for each validation set and the standard deviation of all RMSE values per test set are calculated. This is done for each approach for combination sizes from 1 to 10 and each possible production process parameter combination. Looking at the implementation, most of this evaluation steps are executed in the subroutine `performEvaluationCV`.

Applying the Student's *t*-test. A Student's *t*-test is only capable to compare two approaches. Since we have 5 methods to be evaluated against each other, all possible combinations are considered. The RMSE values achieved through the leave-one-out CV on the validation sets are used as an input to the Student's *t*-test. To reduce complexity, only the production process parameter combinations leading to the best predictions in terms of minimizing the RMSE on the real-test-set are used. This reduces the Student's *t*-test analysis set to a handful combinations per combination size. Finally, the results of the Student's *t*-test are evaluated.

Regarding the implementation, the Student's *t*-test is executed and graphically visualized in `EvaluateResults.m` and interpreted using the subroutine `interpretStudentTResults`. The best results are plotted in the subroutine `plotRMSEoverMethods`.

Summing up, the procedure executed to evaluate Student's *t*-tests:

(1) Create real-test-set and all-test-set.

(2) Perform leave-one-out CV for both sets on each approach using each combination size to determine RMSE and standard deviation.

(3) Select production process parameters leading to the best prediction results on real-test-set by minimizing RMSE.

(4) Perform Student's t-test on best production process parameter set using real-test-set and all-test-set results for each possible approach combination and each combination size.

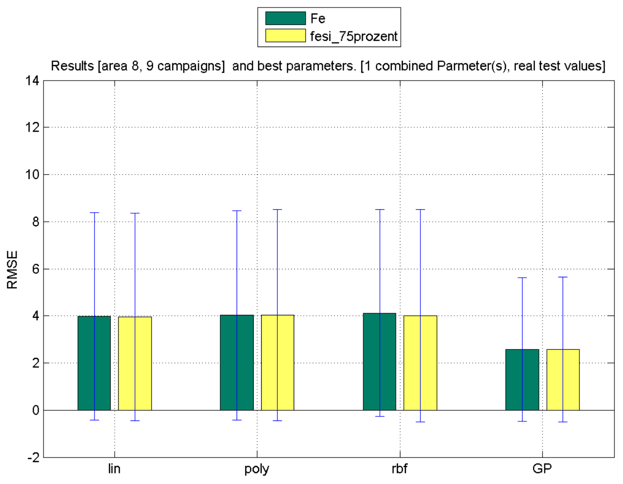
(5) Interpret Student's t-test results to determine best approach in terms of APO.

Interpretation of Student's *t*-test results. Recap, all statistical significance tests are performed using a significance level of 5%. As mentioned earlier, the best results are plotted using the subroutine `plotRMSEoverMethods`. In doing so, the mean RMSE value and the standard deviation of all RMSE values of each approach are plotted for all combination sizes. This is done for the real-test-set and the all-test-set. Figure 5.9 sketches few results from the real-test-set and Figure 5.10 from the all-test-set achieved by analysing area 8. Similarly, Figure 5.11 covers the real-test-set results and Figure 5.12 the all-test-set results gathered from area 99.

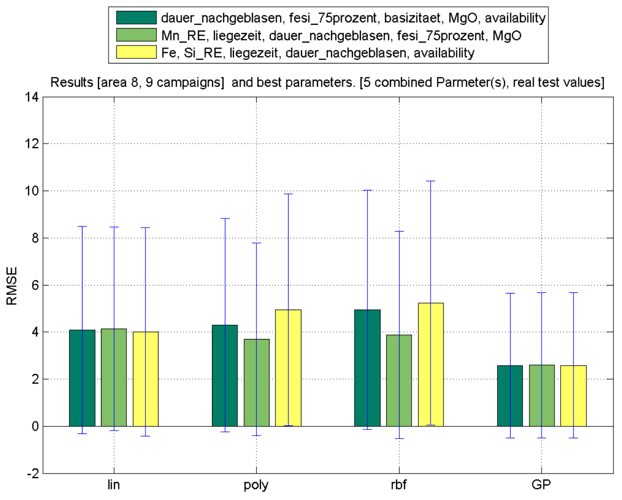
Student's *t*-test results in area 8. Looking at the area 8 results on real-test-set in Figure 5.9 one can see significant differences even for one production process parameter. The *GP* results outperform the other approaches in terms of mean RMSE values and standard deviation. This result does not change by increasing the combination size. For large combination sizes clear differences can be found. *GP* leads to the best performances, ahead of *MLR* while the *SVR* kernels both does not perform that well. Figures 5.9(c) and 5.9(d) reflect those results.

Figure 5.10 focusses on the all-test-set results achieved in area 8. Comparing to the results on the real-test-set it becomes more difficult to find significant differences. By increasing the combination size slightly significant differences can be observed. Starting from an combination size of 2, *GP* significantly performs better than *SVR*. Using combination size 10 as depicted in Figure 5.10(d) *GP* even performs better than the linear model indicated through the smaller standard deviation and the lower mean RMSE. Similar to the real-test-set the linear model leads to better predictions than the time consuming *SVR*.

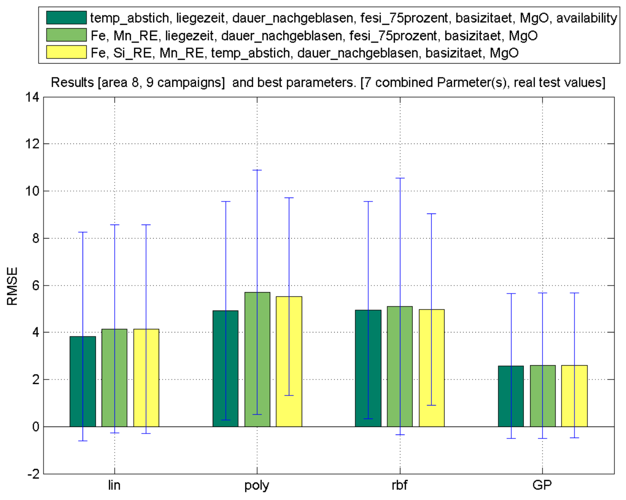
Concluding the results in area 8, the results depend on the test set selection. Depending on the used test set significant performance differences can be found earlier in terms of combination sizes. Nevertheless, in the end *GP* always leads to the best prediction performance results.



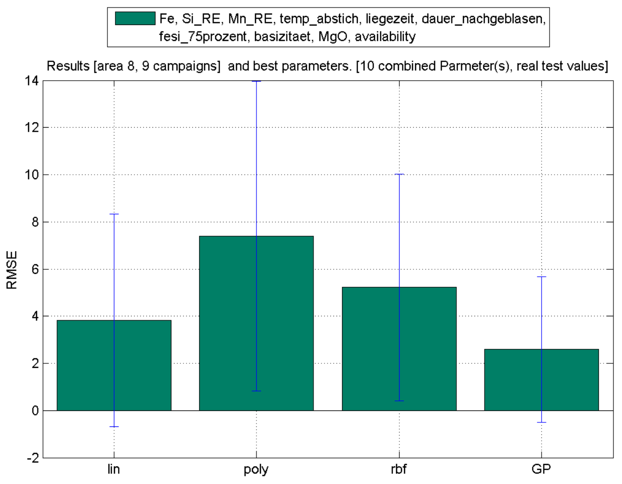
(a) real-test-set, 1 parameter, area 8



(b) real-test-set, 5 parameters, area 8

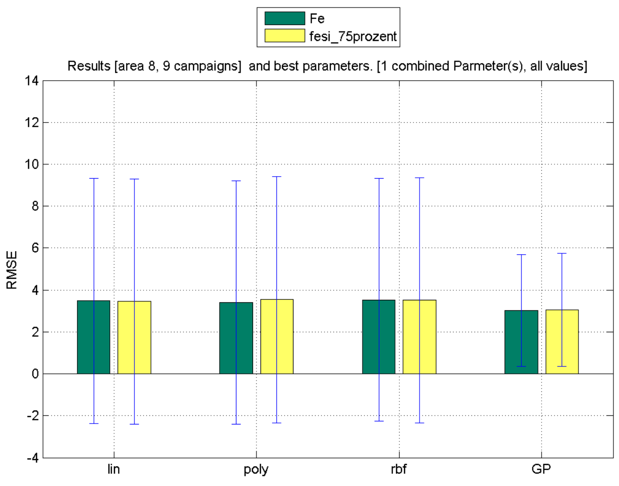


(c) real-test-set, 7 parameters, area 8

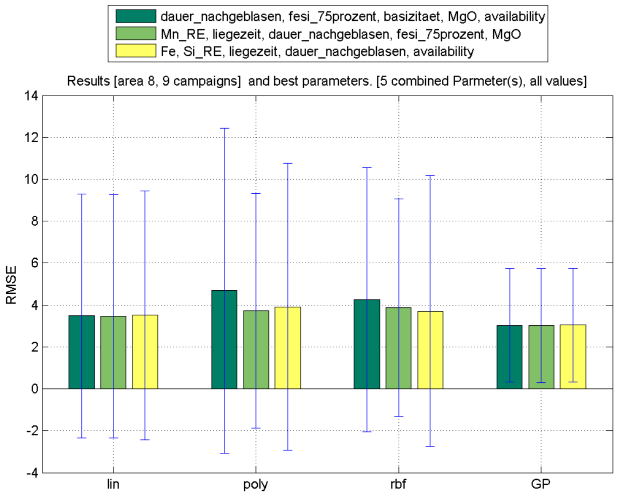


(d) real-test-set, 10 parameters, area 8

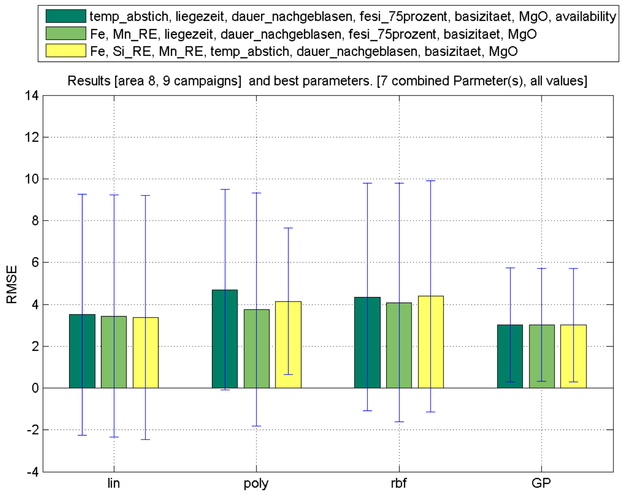
Figure 5.9.: Results of significance tests applied on real-test-set in area 8. Even one production process parameter shown in (a) results in significant differences. *GP* results outperform the other approaches. Increasing the combination size does not change the result. *GP* leads to the best performance, ahead of *MLR* while the *SVR* kernels both does not perform that well indicated in (c) as well as in (d).



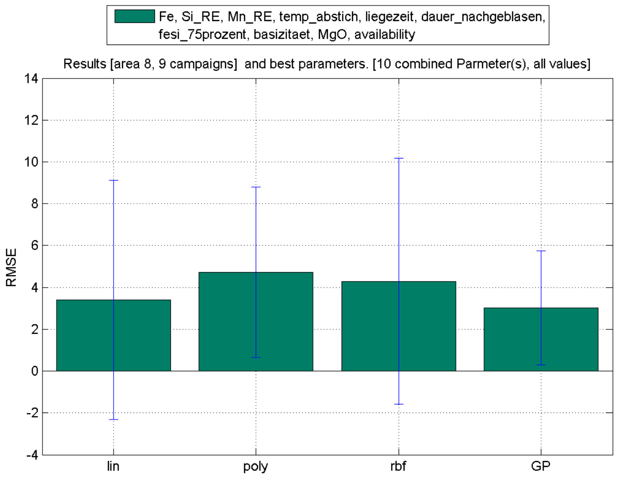
(a) all-test-set, 1 parameter, area 8



(b) all-test-set, 5 parameters, area 8



(c) all-test-set, 7 parameters, area 8



(d) all-test-set, 10 parameters, area 8

Figure 5.10.: Results of significance tests applied on all-test-set in area 8. Compared to the real-test-set it is more difficult to find significant differences. Smaller combination sizes like 1 and 5 shown in (a) and (b) do not indicate clear significant differences. The tendency that *GP* significantly performs better than *SVR* is approved in (d) where 10 production process parameters are combined. *GP* even performs better than the linear model reflected through the smaller standard deviation and the lower mean RMSE. Similar to the real-test-set the linear model leads to better predictions than *SVR*.

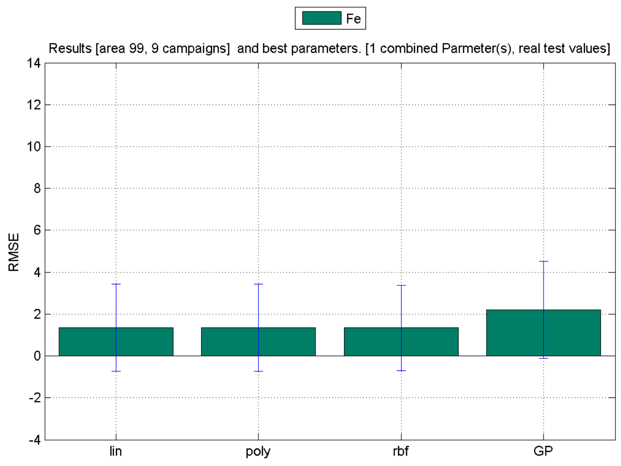
Student's *t*-test results in area 99. Figure 5.11 summarizes a set of results achieved by applying the significance tests to the real-test-set in area 99. Initially, the *GP* performs really bad in terms of mean RMSE as sketched in Figure 5.11(a). Nevertheless, no significant differences are found between the approaches. By increasing the combination size the linear model significantly performs better than the other approaches. No significant differences between *SVR* and *GP* are found so far. This situation does not change until increasing the combination size to 5. Figure 5.11(b) reflects the results for combination size 5. Again the *MLR* outperforms *GP* and *SVR* with RBF kernel. Additionally, the *SVR* with polynomial kernel significantly performs better than *GP*. This momentum changes by increasing the combination size again. As depicted in Figure 5.11(c) the prediction performance of *SVR* decreases for higher combination sizes. Thus, *GP* and linear models significantly perform better than both *SVR* kernels. In the end, by combining all available production process parameters, the linear model significantly performs better than the *SVR* with both kernels and even the *GP* performs significantly better than the *SVR* with polynomial kernel. This is sketched in Figure 5.11(d).

Looking at the results gathered from analysing area 99 on the all-test-set summarized in Figure 5.12, one can see large significant differences. Even for one production process parameter *GP* performs significantly better than the other approaches. Increasing the combination size the significant performance dominance of *GP* results are confirmed. Looking at 10 combined production process parameters sketched in Figure 5.12(d) one can see that *GP* clearly outperforms all other approaches. Moreover, the linear model performs significantly better than the *SVR* approaches.

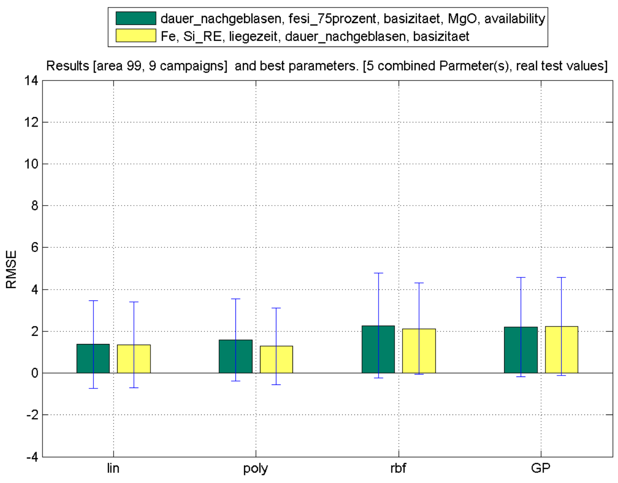
Summing up, significance tests in area 99 again explicitly reflects the difference of results depending on the test set selection. Moreover, the combination size has influence on the significance results as well. Concluding, the linear model and the *GP* clearly perform better than the *SVR* approach.

Comparison to practical experience. Finally, the achieved results are reviewed in terms of practical experience. Experienced RHI AG specialists know that refractory material wears 0.2 to 0.7 millimetres per heat depending on the area in the converter. Looking at the results in area 99 sketched in Figure 5.12 the mean RMSE of the *GP* approach lies around 0.7. That means that the APO prediction in average produces an error of factor 2 in area 99. Results in area 8 are not that accurate due to various reasons mentioned earlier. In conclusion, RHI AG specialists are quite satisfied with the prediction results with respect to the limitations of APO.

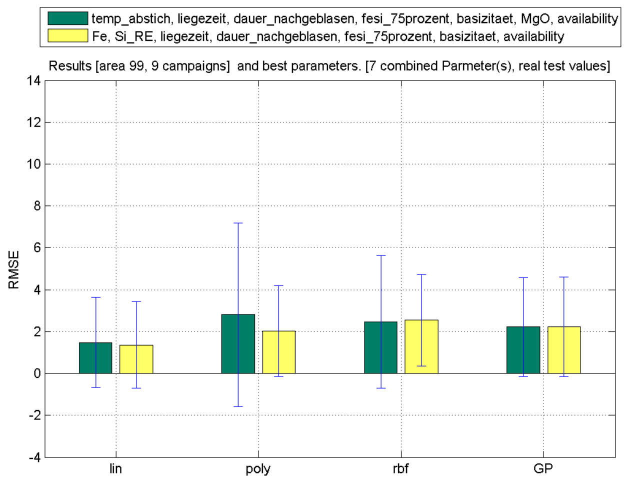
Summing up, statistical tests are used to determine significant differences between the applied approaches with respect to the prediction performance. Different test sets are investigated and analysed on combination sizes from 1 to 10 in area 8 and area 99.



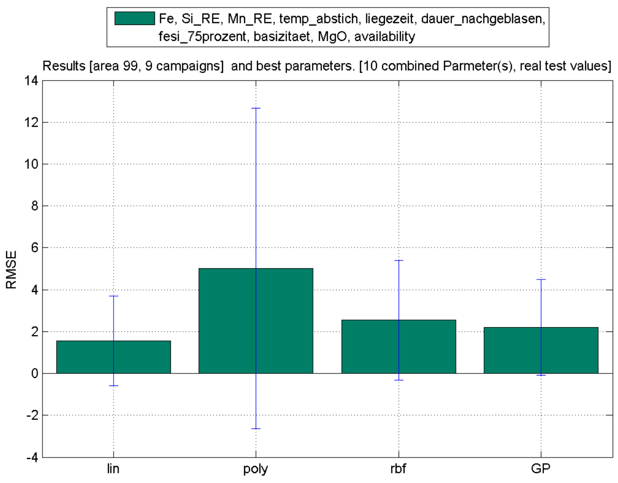
(a) real-test-set, 1 parameter, area 99



(b) real-test-set, 5 parameters, area 99

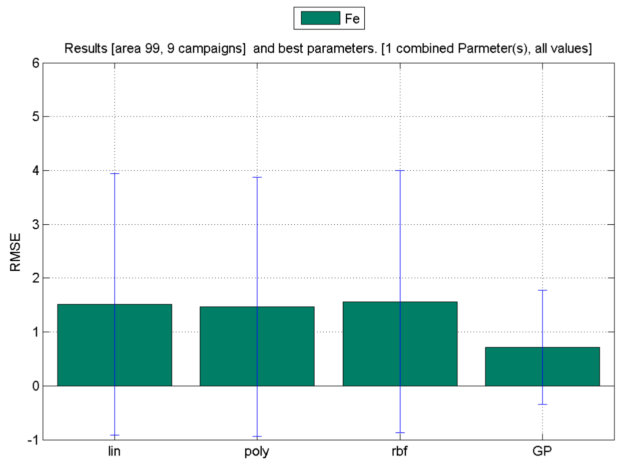


(c) real-test-set, 7 parameters, area 99

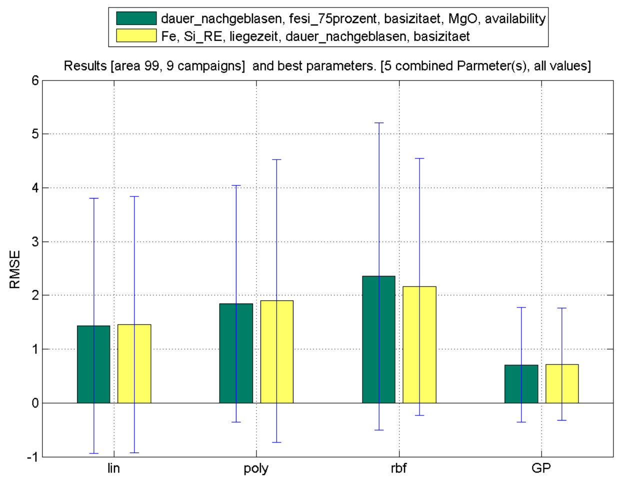


(d) real-test-set, 10 parameters, area 99

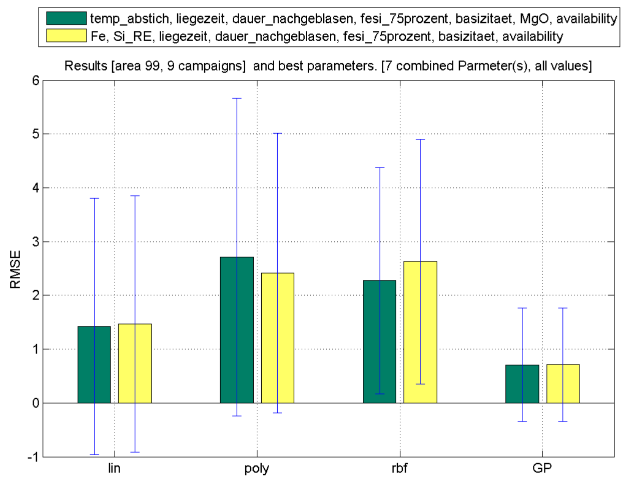
Figure 5.11.: Results of significance tests applied on real-test-set in area 99. (a) shows that *GP* performs really bad, but no significant differences can be found, initially. For combination size 5 in (b) interesting results can be observed. *MLR* outperforms *GP* and *SVR* with RBF kernel. Additionally, the *SVR* with polynomial kernel significantly performs better than *GP*. Increasing the combination size to 7 as shown in (c) changes this momentum. The prediction performance of *SVR* decreases and both *GP* and linear models significantly perform better than both *SVR* kernels. By combining all available production process parameters in (d), the linear model significantly performs better than the *SVR* with both kernels and even the *GP* performs significantly better than the *SVR* with polynomial kernel.



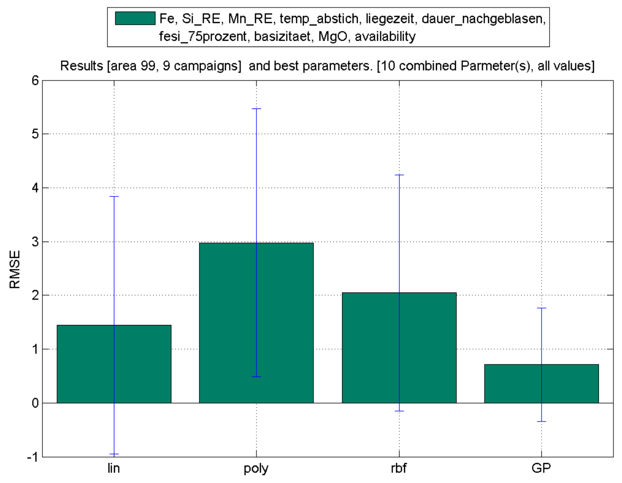
(a) all-test-set, 1 parameter, area 99



(b) all-test-set, 5 parameters, area 99



(c) all-test-set, 7 parameters, area 99



(d) all-test-set, 10 parameters, area 99

Figure 5.12.: Results of significance tests applied on all-test-set in area 99. Significant differences can be observed for all combination sizes. Even for one production process parameter in (a) *GP* performs significantly better than the other approaches. Increasing the combination size the significant performance dominance of the *GP* results are confirmed as shown in (b) and (c). Applying combination size 10 as shown in (d) one can see that *GP* clearly outperforms all other approaches while the linear model performs significantly better than the *SVR* approaches.

5.3. Conclusion

Concluding this chapter, error metrics and methods to evaluate the APO approaches are introduced. Then, the importance of the data set size is mentioned. It is shown, that bigger data set sizes improve the prediction results explicitly. Additionally, the hyper-parameter training results of each approach are covered in terms of evaluation time and precision. Most of the results are plotted using REC curves. In the end, the *GP ARD* results are discussed showing the influence of each production process parameter on the refractory wear. Finally, all APO approaches are evaluated against each other to find the best suitable approach. This is done by applying statistical tests. APO applied Student's t-tests to achieve deeper insight to the performance of each approach.

Altogether, significant differences occur in dependence of the analysed area, the combination size and the used test set. Due to the small APO real-test-set a further test set including all data is investigated. In doing so, a more general performance distinction is possible. As a result, *GP* perform mostly well. In addition, *GP* offers the interesting ARD feature to extract the weighted production process parameter influences on the refractory wear. Moreover, it is shown that the linear model performs better than both *SVR* kernels.

Teil VI.

CONCLUSIONS

6 Conclusions

Contents

This chapter briefly summarizes all results gathered in this thesis in Section 6.1 and provides a outlook for future work in Section 6.2.

6.1. Conclusions

This thesis deals with tasks in steel manufacture. Although it is a rather old business, steel is still required in huge amounts in various domains. Therefore, steel-plant companies and their suppliers are interested in optimizing routines and processes on the manufacturing site of a steel-plant. Thus, RHI AG carried out this thesis to analyse the relationships between production process parameters defining the steel-making process and the refractory wear occurring inside a converter.

This research made use of different machine learning approaches to analyse these relationships. Linear ones like *Linear Regression* (*LR*) or *Multiple Linear Regression* (*MLR*) are considered as well as non-linear techniques like *Support Vector Regression (SVR)* with RBF or polynomial kernels. Moreover, *Gaussian Processes* (*GP*) is an additional method applied in this thesis.

The data sets provided by the cooperating steel-plant HKM in Duisburg are delivered in unusable formats for automatic processing. Thus, a pre-processing step for both production process parameters and laser measurements is required. This thesis introduced various simplifications to make the definition of prediction models as simple as possible and to reduce computational complexity. Among others, simplifications like reducing the production process parameter subset to 10 selected parameters or representing data through characteristic values are mentioned. All analyses are done for two converter areas and all possible combinations of production process parameters within the combination size range from 1 to 10. It is shown that data set size has tremendous impact on the prediction performance. Increasing data amounts clearly lead to improved results.

Concluding the different approaches' results, it is shown that even simple *MLR* models are able to provide some insight to the relationships between production process parameters and refractory wear. The non-linear *SVR* approaches using RBF and polynomial kernels both require a time consuming hyper-parameter training step. In advance, 70% of the data set are randomly chosen as training set and the remaining set acted as a test set. An iterative grid search is introduced guaranteeing a training speed-up of almost 70%. The *SVR* models are trained using the found hyper-parameters and their predictions tested using the test set. It is shown that *SVR* results improves up to a certain combination size. In fact, *MLR* results are more accurate than both *SVR* kernels. In addition, the prediction performance in area 99 is better compared the one achieved in area 8. Like for *SVR*, *Gaussian Processes* require all the optimal hyper-parameters to be determined, in this case using a conjugate gradient-based optimization concept. Predictions made by a *GP* model, trained using the found hyper-parameters, show that, by tendency, the *GP* perform better than the non-linear *SVR* results. While *MLR* showed superior performance with respect to the evaluation time, *GP* still converges faster than the time-consuming *SVR* approaches.

Since the available data set is quite small, this thesis used statistical tests to evidently evaluate the different approaches against each other to find the best suitable approach. Particularly, Student's t-tests including cross validation are applied on two different test sets to determine significant differences. On the one hand, the real-test-set is used, on the other hand, due to the small real-test-set, the whole data set is used as so-called all-test-set. Using the all-test-set including all data a more general performance distinction can be made. Statistical tests in this thesis have shown that *GP* perform mostly well. In contrary to the other approaches, *GP* offer the automatic relevance determination feature to extract the specific production process parameter impact on the refractory wear. Moreover, it is shown that the linear model clearly performs better than both *SVR* kernels. Finally, this thesis demonstrated that prediction performance depends on the selected production process parameters, the combination size and the analysed area as well.

6.2. Future Outlook

As mentioned earlier, this thesis made use of various restrictions and simplifications, once more summarized for a better overview:

(1) only 10 out of 100 production process parameters analysed

(2) lots of production process data represented through a characteristic value due to too less laser measurements

(3) no maintenance data included

(4) no refractory material data included

(5) analyses performed on converter area definitions based on experience

(6) interpolation and consolidation strategies applied to ensure consistent data set.

To further improve the prediction performance of the APO approaches the amount of simplifications has to be decreased. In doing so, an inclusion of maintenance data and refractory data would be helpful. Additionally, considering more or even other production process parameters can lead to better results. The converter areas of interest are selected based on experience. One could even investigate in clustering algorithms to group areas with similar characteristic behaviour automatically.

As the data sets are increased vastly only towards the end of the project, *Artificial neural networks (ANNs)* are not part of the analyses in this thesis. Therefore, further investigations reasonably include *ANNs* to assess whether its prediction capabilities could outperform those of the covered approaches.

As one can see, this topic still remains a research topic offering lots of opportunities to improve the results gathered in this thesis.

Teil VII.

APPENDIX

A Appendix

A.1. Listings

REC Plot Algorithm. The following plot algorithm uses $\epsilon_i = loss(f(x_i), y_i), \quad i = 1, \ldots, N$ as input. It is assumed that errors ϵ_i are sorted in ascending order. The plot command interpolates the plotted points with a line. Listing A.1 sketches the REC plot algorithm implemented in the MATLAB® function `rec_curve.m`.

```
$\epsilon_{prev}$ := 0; correct := 0;
for i = 1 to m
    if $\epsilon_i$ > $\epsilon_{prev}$ then
        plot($\epsilon_{prev}$, $correct/m$)
        $\epsilon_{prev}$ := $\epsilon_i$
    end
    correct := correct + 1
end
plot($\epsilon_m$, $correct/m$)
```

Listing A.1: Pseudo code of REC plot algorithm. (Source: [Bi and Bennett, 2003])

A.2. Installation requirements

MEX prerequisites and installation details. In order to use MEX-files properly the following prerequisites have to be fulfilled:

- A proper compiler (C, C++, Fortran) needs to be installed
- A MEX function is necessary to build MEX-files.

MATLAB® provides a C compiler, Lcc but since the MATLAB® interface of LIBSVM is implemented in C++ a C++ compiler is necessary. Hence, Microsoft Visual Studio C++ 2010 Express has to be installed. To ensure that all functions work properly the SDK is installed as well. Since MATLAB® version 2009b is used an additional patch is necessary to make the newer Visual Studio C++ 2010 Express visible. For this reason the patch from MathWorks[1] is downloaded and installed. Using `mex -setup` in MATLAB® the correct compiler has to be selected as shown in Listing A.2 in detail.

Finally, the usage of LIBSVM has to be ensured by switching to the LIBSVM directory and typing the `make` command in the MATLAB® console. Further tests are shown in the LIBSVM readme. After installing MEX, LIBSVM functions can be used directly in MATLAB® like built-in MATLAB® functions.

```
mex -setup
Please choose your compiler for building external interface (MEX) files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2009b\sys\lcc
[2] Microsoft Visual C++ 2010 Express in C:\Program Files\Microsoft Visual Studio 10.0
[0] None

Compiler: 2

Please verify your choices:
Compiler: Microsoft Visual C++ 2010 Express
Location: C:\Program Files\Microsoft Visual Studio 10.0

Are these correct [y]/n? y

***************************************************************************
  Warning: MEX-files generated using Microsoft Visual C++ 2010 require
           that Microsoft Visual Studio 2010 run-time libraries be
           available on the computer they are run on.
           If you plan to redistribute your MEX-files to other MATLAB
           users, be sure that they have the run-time libraries.
***************************************************************************

Trying to update options file: C:\Users\manuel\AppData\Roaming\MathWorks\MATLAB\R2009b\mexopts.bat
From template:              C:\PROGRA~1\MATLAB\R2009b\bin\win32\mexopts\msvc100freeopts.bat

Done . . .

**************************************************************************
  Warning: The MATLAB C and Fortran API has changed to support MATLAB
           variables with more than 2^32-1 elements.  In the near future
           you will be required to update your code to utilize the new
           API. You can find more information about this at:
           http://www.mathworks.com/support/solutions/data/1-5C27B9.html?solution=1-5C27B9
           Building with the -largeArrayDims option enables the new API.
**************************************************************************
```

Listing A.2: Installation instruction for selecting the correct MEX compiler

[1] http://www.mathworks.com/support/solutions/en/data/1-D5W493/?solution=1-D5W493

Installing `GPML toolbox` for `MATLAB`®. Setting up the `GPML toolbox` for `MATLAB`® is quite easy. Once the `GPML toolbox` is downloaded from the GPML website[2] one has to unpack the archive. In doing so, the 6 subdirectories `cov`, `doc`, `inf`, `lik`, `mean` and `util` can be found. It is not necessary to install anything, only a execution of `startup.m` script is required to set the path.

[2] http://www.gaussianprocess.org/gpml/code/matlab/doc/

Teil VIII.

BIBLIOGRAPHY

Bibliography

[Bi and Bennett, 2003] Bi, J. and Bennett, K. P. (2003). Regression error characteristic curves. In *Proceedings of the 20th International Conference on Machine Learning*, pages 43–50.

[Bishop, 2007] Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition.

[Bolbrinker and Verein Deutscher Eisenhüttenleute, 1992] Bolbrinker, A.-K. and Verein Deutscher Eisenhüttenleute (1992). *Steel manual*. Verein Deutscher Eisenhüttenleute (VDeh), Düsseldorf, Germany. English translation of the German 'Stahlfibel'.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[Chen and Wang, 2007] Chen, K.-Y. and Wang, C.-H. (2007). Support vector regression with genetic algorithms in forecasting tourism demand. *Tourism Management*, 28(1):215 – 226.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20(3):273–297.

[Dorff et al., 2010] Dorff, K. C., Chambwe, N., Srdanovic, M., and Campagne, F. (2010). BDVal: reproducible large-scale predictive model development and validation in high-throughput datasets. *Bioinformatics (Oxford, England)*, 26(19):2472–2473.

[Farrell and Correa, 2007] Farrell, M. and Correa, A. (2007). Gaussian process regression models for predicting stock trends. *Relation*, 10:3413.

[Fisher, 1925] Fisher, R. A. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh.

[Grauman and Darrell, 2005] Grauman, K. and Darrell, T. (2005). The pyramid match kernel: discriminative classification with sets of image features. In *Computer Vision,* *2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1458 – 1465 Vol. 2.

[Gunn, 1998] Gunn, S. (1998). Support vector machines for classification and regression. Technical report, Faculty of Engineering, Science and Mathematics, School of Electronics and Computer Science.

[Hanke et al., 2009] Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., and Pollmann, S. (2009). PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data.

[Hsu et al., 2010] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2010). A practical guide to support vector classification. *Bioinformatics*, 1(1):1–16.

[Hüttenwerke Krupp Mannesmann, 2002] Hüttenwerke Krupp Mannesmann (2002). HKM - Steel: That's our business. 2002, Duisburg.

[Jandl, 2009] Jandl, C. (2009). Influences of BOF refractory wear. Internal Presentation, RHI AG, Vienna, 2009.

[Jandl, 2011] Jandl, C. (2011). Refractory Know-How - Blast Furnace to Basic Oxygen Furnace. Guest Lecture at University of Leoben.

[Lammer, 2011] Lammer, G. (2011). Autonome Pflegeprogrammoptimierung (APO, stands for intelligent maintenance optimization) - Outline. May 31st 2011, RHI AG.

[Li et al., 2000] Li, Y., Gong, S., and Liddell, H. (2000). Support vector regression and classification based multi-view face detection and recognition. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 300 –305.

[Lu et al., 2009] Lu, C.-J., Lee, T.-S., and Chiu, C.-C. (2009). Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, 47(2):115 – 125.

[Maass, 2008] Maass, W. (2008). Computational intelligence lecture notes.

[MATLAB, 2011] MATLAB (2011). *Version 7.13 (R2011b)*. The MathWorks Inc., Natick, Massachusetts.

[MINTEQ International GmbH - FERROTRON Division, 2009] MINTEQ International GmbH - FERROTRON Division (2009). LaCam system measuring principle and general functions. 2009, Duisburg.

[Montgomery and Peck, 1992] Montgomery, D. and Peck, E. (1992). *Introduction to linear regression analysis*. Wiley series in probability and mathematical statistics. Wiley, New York, NY [u.a.], 2. ed edition.

[Mori and Kurata, 2008] Mori, H. and Kurata, E. (2008). Application of gaussian process to wind speed forecasting for wind power generation. In *Sustainable Energy Technologies, 2008. ICSET 2008. IEEE International Conference on*, pages 956 –959.

[Nickerson, 2000] Nickerson, R. S. (2000). Null hypothesis significance testing: a review of an old and continuing controversy. *Psychological Methods*, 5(2):241–301.

[Nivre et al., 2007] Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

[Rasmussen and Nickisch, 2010] Rasmussen, C. E. and Nickisch, H. (2010). Gaussian processes for machine learning (gpml) toolbox. *Journal of Machine Learning Research*, 11(9):3011–3015.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[RHI AG, 2011a] RHI AG (2011a). Sauerstoffkonverter - Basic Oxygen Furnace. METEC Edition 2011, Wien.

[RHI AG, 2011b] RHI AG (2011b). RHI Bulletin - The journal of refractory innovations. volume steel edition. RHI AG, METEC - 8th International Metallurgical Technology Trade Fair with Congresses.

[RHI AG, 2012] RHI AG (2012). RHI AG - A world leading refractories technology. Roadshow Presentation 2012, Wien.

[Ricanek and Tesafaye, 2006] Ricanek, K. and Tesafaye, T. (2006). MORPH: A Longitudinal Image Database of Normal Adult Age-Progression. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 341–345, Southhampton UK. IEEE Computer Society.

[Rugg, 2007] Rugg, G. (2007). *Using Statistics*. Open University Press.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14:199–222.

[Tay and Cao, 2001] Tay, F. E. H. and Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317.

[Trafalis and Ince, 2000] Trafalis, T. and Ince, H. (2000). Support vector machine for regression and applications to financial forecasting. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 348 –353 vol.6.

[Wen et al., 2009] Wen, Y., Cai, C., Liu, X., Pei, J., Zhu, X., and Xiao, T. (2009). Corrosion rate prediction of 3c steel under different seawater environment by using support vector regression. *Corrosion Science*, 51(2):349 – 355.

[World Steel Association, 2012] World Steel Association (2012). Overview of the steelmaking process - A fold-out poster.

[Xue and Titterington, 2011] Xue, J.-H. and Titterington, D. (2011). t -tests, f -tests and otsu's methods for image thresholding. *Image Processing, IEEE Transactions on*, 20(8):2392 –2396.

[Yuan and Huang, 2004] Yuan, Z. and Huang, B. (2004). Prediction of protein accessible surface areas by support vector regression. *Proteins*, 57(3):558–564.

[Zhang and Yeung, 2010] Zhang, Y. and Yeung, D.-Y. (2010). Multi-task warped gaussian process for personalized age estimation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2622 –2629.

[Zhang et al., 2011] Zhang, Z., Tong, T., and Song, K. (2011). A novel gpls-gp algorithm and its application to air temperature prediction. In *Natural Computation (ICNC), 2011 Seventh International Conference on*, volume 3, pages 1445 –1449.