Università degli studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Data analysis techniques for predicting and optimizing an industrial process

Relatore
Prof. Fabio Vandin

Laureando
Dario Simionato
matricola 1179812

9 Dicembre 2019
Anno accademico 2018/2019

Abstract

Data is fundamental in order to have a deep understanding of a complex scenario, whether it is the financial health of a business, specific market trends, influence of given policies, or performances of a process. Its key role is to highlight specific objective aspects in order to create a big picture that summarizes an activity helping in decision making tasks. Given that misinterpreted data may lead to wrong choices while complete and accurate information can take to great benefits, there is no surprise in the increase of the role of data-driven decisions in all areas. A key factor of this revolution has been played by technology, since nowadays it is affordable to collect and store big data quantities and hardware improvements made complex machine learning tasks feasible.
This thesis studies the influence of data analysis and machine learning applied to the industrial productive process of wheels casting in which it has been tried to predict product quality using machine settings and in-process measurements. It can be considered as a preliminary feasibility research of the potential of such approaches in process analysis and control. The working method used resembles black box modeling in which a system is described uniquely as a set of inputs and outputs since its creation needed a very little domain expertise and it did not require any process knowledge to learn how to predict wheel quality. Despite that, the underlying techniques used have been described exploiting a statistical framework that helped in the introduction of key concepts for data evaluation and results interpretation.

Contents

List of Figures

1

The productive process

In this chapter RONAL group is presented and the wheel productive process is briefly described.

1.1 RONAL group

RONAL group is one of the world's leading manufacturers and suppliers of light alloy wheels for passenger cars and commercial vehicles*. The company counts over 8000 employers throughout the world. In 2018 it sold over 21 million wheels for a revenue of 1.4 billion euros. The group was founded in 1969 in Forst (Germany) and in 6 years it expanded in France. Between 80s and 90s RONAL opened productive plants in Spain, Portugal, Poland, Mexico and Czech Republic and only recently, in 2007, it acquired Speedline SRL in Santa Maria di Sala (Italy) due to its expertise in flowforming technology.

Figure 1.1: RONAL group logo.

*Further information at https://www.ronalgroup.com/

1.2 The productive process

The productive process is divided in 5 phases:

- Alloy fusion, wheel casting and X-ray inspection;
- Flowforming and thermical treatments;
- Mechanical processing;
- Finishing and painting;
- Quality tests.

After each phase the wheel is visually checked and, in case of defects, it is discarded.
The process starts with an alloy fusion of aluminum, silicon and magnesium (AlSi7Mg0.3) on different ovens. Raw materials are composed by alloy ingots and processing wastes such as chips and scrap wheels. Fused alloy is then carried to low pressure casting machines (BP from the italian name *"bassa pressione"*) each one having an oven for maintaining the temperature of the alloy as depicted in Figure 1.2.
Wheel casting process is divided into 4 different phases shown in Figure 1.3:

1. At first air is injected in the oven in order to increase pressure. Melted alloy goes up the ascent tube.
2. As soon as the ascent tube is filled up, the filling phase of the mould starts. The speed of this process is variable and it depends on a filling rate parameter.
3. When the mould is filled up, pressure is maintained stable for a certain amount of time in order to compact the wheel. Multiple cooling circuits cool the wheel using air or a mixture of air and water
4. At the end of phase 3 the mould is closed, pressure is released and cooling continues until the wheel solidifies.

This procedure lasts from 240 to 420 seconds depending on wheel size and on filling or cooling specifications.

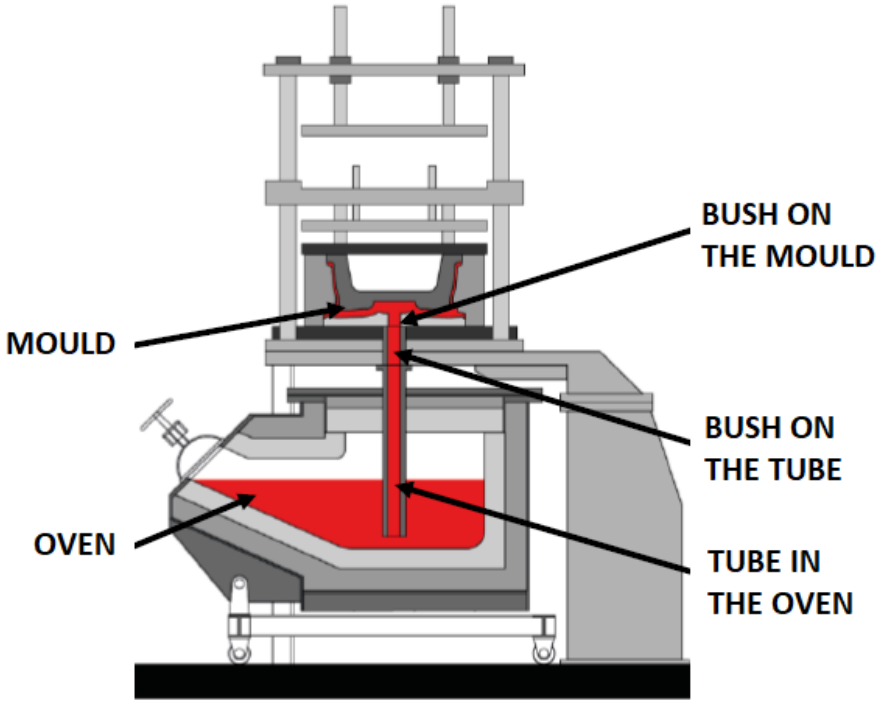

Figure 1.2: Schematic plot of the casting machine. The oven contains fused alloy that rises the tube in order to fulfill the mold. After wheel casting, the mould is opened and the released wheel sent to a conveyor.

The wheel is then moved to a conveyor which sends it to a machine that removes the risers (*materozze* in Italian) and then analyzes it using X-rays in order to find impurities and check its density. This procedure is machine-aided since employees classify only scanned wheels whose images differ significantly from a referral one. If X-ray analysis didn't show any criticity, the wheel is sent[†] to flowforming

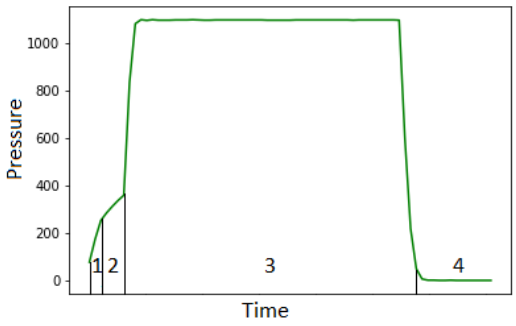

Figure 1.3: Oven pressure over time: the 4 different phases are highlighted.

[†]Monoblock wheels skip flowforming since their casting shape is already the final one.

department where it is flow formed in order to increase its mechanical performances while maintaining a low weight. Some thermical processes are performed on the wheel that is then processed mechanically in order to remove metal smudges and to create the hole for the valve. Finally the wheel is painted and prepared for the shipping.
During the whole production chain several control quality checks are applied to the wheel in order to ensure its safety and the meeting of client's requests.

1.3 Wheel characteristics

In Figure 1.4 are shown the 3 parts that characterize a wheel:

- The hub (*mozzo* in Italian) is the central part of the wheel and the one from which the alloy rises in order to fulfill the mould.
- The spoke (*razze* in Italian) is the most visible part of the wheel and it links the hub to the rim.
- The rim (*canale* in Italian) is the largest part of the wheel and the one on which the tire is placed.

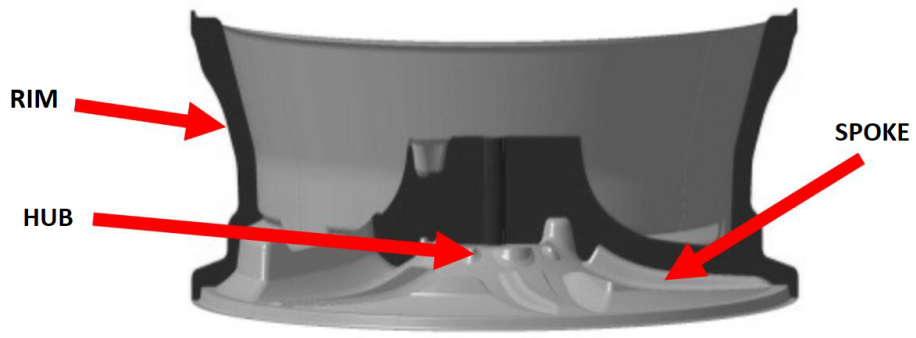

Figure 1.4: Vertical section of a wheel.

As introduced before, wheels may be divided into monoblock and flowformed. Monoblock wheels shape does not change after the casting while flowformed

wheels are subject to a procedure that enlarges the rim via mechanical operations. This latter category of wheel is more performing and lighter.
In the thesis 4 wheels have been analyzed but, due to the amount of data received, only three of them have been effectively studied. They all are big flowformed wheels (20 inches or more) and their internal code are: 3213, 3701, 4273 and 4509. Figure 1.5 shows them.

Figure 1.5: Image of the wheels analyzed in this case study.

2

Data structure and analysis

In this chapter dataset structure and the main projectual choices have been described. The aim of this study is to exploit data from wheel casting and X-ray analysis in order to predict whether a wheel will be OK or discarded (NOK). The choice of limiting this analysis to only these two processes is due to logistic reasons: after X-ray analysis, wheels are removed from the conveyor and it is not possible to track them anymore.
Data analysis and prediction have been perfomed by programming in python 3 and exploiting jupyter notebook to divide code into cells and to show results nicely. Mathematical operations had been performed using numpy library while prediction models are part of sklearn.

2.1 Dataset structure

Wheel information are spreaded into 4 .csv files of variable length*:

- "PRD_XXXX.csv" contains general wheel casting information such as the project ID number, the mould ID code, casting machine ones and casting

*Each file contains a timestamp of the saving date in its name. In the following examples it will be referred as "XXXX".

process result.

- "PRM_XXXX.csv" contains casting setups such as the filling rate of the mould, a setting for the desired oven temperature, the final pressure of the working cycle and the cooling systems settings.

- "PD_XXXX.csv" contains measurements of physical features (mostly temperatures and pressures) during the casting process. Each measurement is taken every 5 seconds so for each wheel multiple values are recorded.

- "XRAY_XXXX.csv" contains X-ray scan information such as the number of the X-ray machine that performed the analysis and its result.

Each tuple of every file is univocally referred by the combination of a serial number and a date that has been used as a key to get data of a specific wheel[†]. Appendix A contains further information on the different parameters of each file and their meaning.

2.2 Data preprocessing

Saving data in different files is convenient but some troubles may arise in case of writing errors. The most frequent issue is on duplicate tuples since in some cases files have multiple records referring erroneously to the same wheel[‡]. The dual problem of missing parameters has also been found in the data. This behavior may be an issue[§] or not[¶] depending on data values and on which value is missing. Similarly, it has been noticed that some wheels don't have enough casting process measurements. Casting process lasts at least 4 minutes (which means at least 4*60/5=48 casting process measurements) but some of them are produced in a shorter amount of time: these wheels are used only to warm the mould after

[†]This choice is due to the serial number range that is limited to the first 10000 numbers. By doing so scalability has been achieved since a single BP machine produces less than 10000 wheels per day. In case of an extension of this study to multiple BPs this constraint is no longer valid but adding the machine number to the key should solve the issue.

[‡]Some examples may be wheels with duplicate machine settings or non unique X-ray results.

[§]For example if casting settings are missing.

[¶]Think about a scrap wheel after casting: since the wheel is discarded before X-ray analysis, it is obvious not to have its record on X-ray data file.

its change and they are always discarded.

By analyzing the temporal series of process measurements other issues have been noticed. The first one is about temperature measurements of thermocouple number 1 (TC_1) since there have been many cases in which this sensor has been activated but it hasn't been placed correctly on the mould so its values, despite having some physical meaning, they were logically erroneous. In particular, those values were around 40℃ so the thermocouple measured the temperature in the proximity of the casting machine instead of the one of the wheel during casting (that is on average higher than 200℃). Dually some wheels show a very high measurement (higher than 3000℃) and that's the typical behavior of when the thermocouple is detached. This may not be a problem if the wheel doesn't use that specific sensor (some moulds have only place for one or two thermocouples) but a consistency issue arises if sometimes it is used and sometimes not. Another observation has been made on the behavior of humidity parameter instability and this suggests a malfunctioning of its sensor or on the acquisition of its data. Figure 2.1 sums up all the problems presented so far regarding time series analysis.

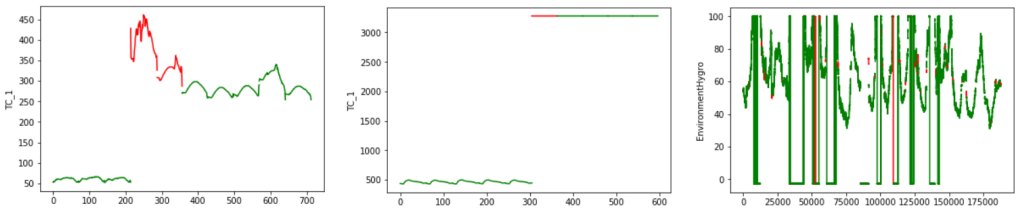

Figure 2.1: Plots of temporal series issues: thermocouple misplaced (left), detached (center) and humidity values highly unstable (right). Red lines represent NOK wheels while green is used for OK ones.

Wheels have been then summarized using a list of values as explained in details later in Section 2.3 and histograms of distributions for each parameter had been analyzed. This study highlighted that some minimum values of pressure measurements, despite referring to non negative physical entities, they were stored with negative values due to some data collection issue.

In order to deal with all these problems, it has been decided to divide the wheels into 3 categories:

- **Conform wheels** are the subset of wheels whose casting process data meets the two following constraints in order to be analyzed by the program:
  - Casting process should last more than 4.30'.
  - Measures from TC_1 should be on average higher than 200°C.
  - Minimum values of pressure measurements shouldn't be negative.
- **Complete wheels** are a subset of conform wheels that contain all the information needed for prediction. In particular, complete wheels must have their machine setup, their casting measurements and a result that describes their status. A wheel is considered as OK if both casting process and X-ray analysis were positive while it is considered as NOK if casting process or X-ray analysis were unsuccessful[‖].
- **Unique wheels** are those complete wheels whose values were not duplicated during writing phase. It has been decided to study this set since it is unbiased from decisions about which values to keep and which to discard[**].

It is relevant to note that both complete and unique wheels pass an additional filtering stage. If the rules applied to conform wheels consider only one wheel, this procedure has been thought for implementing rules that use the whole data distribution such as distribution-wise average, standard deviation and other statistical features. Currently this stage filters thermocouple values implementing an adaptive strategy to remove automatically:

- Wheels for which the thermocouple should have been attached but it has not by analyzing if at least 15% of the wheels have a valid thermocouple measurement.

[‖]From this analysis it has been decided to exclude also those wheels whose casting process was successful but no X-ray result has been collected since a good casting result is not sufficient to ensure a positive outcome on X-ray inspection. This is the case of those wheels that have been manually removed from the conveyor so it was impossible to track them during X-ray analysis.

[**]In the dataset it has been recorded a wheel with two X-ray analysis, the first one that marks the wheel as scrap and a second with a positive result. Which one to keep? One may suggest to keep the last record but it has been decided to follow a more conservative approach and to not consider that wheel.

- Outliers that differ by more than 10 standard deviations from the mean.

Both control rules have been set empirically by knowing the usual process values and the general trend of the data.

2.3 Feature Engineering

Wheels are then processed in order to create a data vector representing them and summarizing their values. Since there is no standard way to deal with temporal series and developing a new one is out of the scope of this work, it has been decided to summarize process measurements temporal series by calculating their minimum, maximum, average, median and standard deviation. These statistical descriptors have been stacked with casting parameters creating an array of values that represent numerically the wheel.

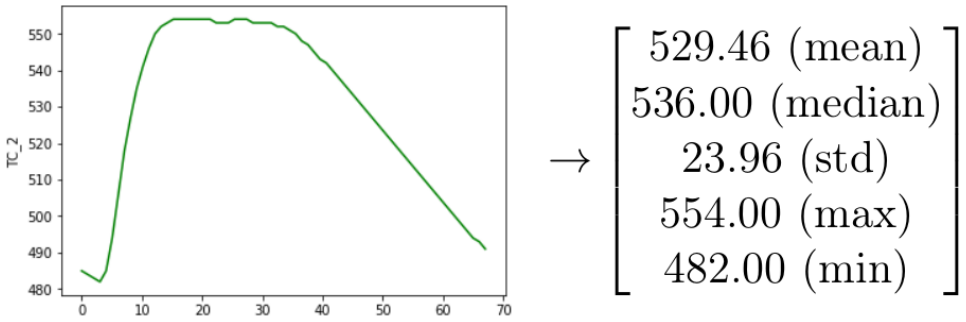

Figure 2.2: Example of data extraction from thermocouple time series. This procedure is repeated for each process measurement each one returning 5 values.

Process measurements taken into account in the creation of the vector may change. In particular, due to an expansion in the data collection system, it is possible to use a standard set of 23 parameters or an extended one of 42. If present, humidity had been discarded given its instability. A particular focus has been given to the parameter "OvenDoorOpen" of the extended analysis set that is equal to 1 if the machine's oven door is opened, 0 otherwise. Given the boolean nature of this parameter and some previous analysis[††] it has been decided to use it to

[††]They shown that there is an higher probability of obtaining a scrap wheel in proximity of alloy refuelings.

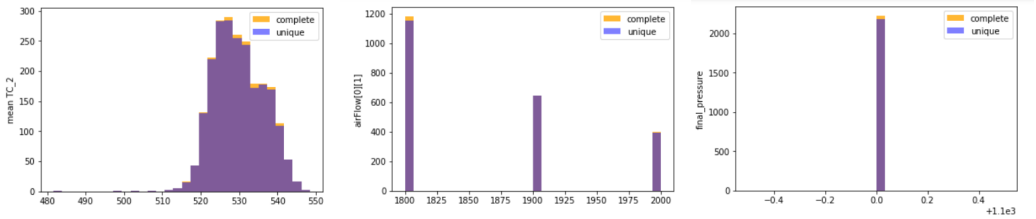

Figure 2.3: Different data distributions in histograms analysis. Both complete and unique wheels have been shown in order to check the absence of biases in selection of unique ones.

calculate the number of wheels casted with success after the last door opening instead of treating it as all the others process measurements. Their complete list and meaning is available in Appendix A.3 while a full description of all the data used to create the vector is available in Appendix A.

2.4 Data distribution and visualization

In order to ensure high data quality, a last study on data has been performed by displaying the histograms of data distribution and time series after filtering procedure. Histogram analysis, as represented in Figure 2.3, shows the variety of data trend behaviors in these plots: some parameters distribute as a Gaussian, others as a multiclass distribution and some are constant. Since these latter values don't change over wheels they do not help in discrimination of OK and NOK wheels and for this reason they have not been considered in the prediction task in order to have lighter vectors and speeding up the computations. Those parameters are usually about constant machine settings or unused thermocouples (5 of them are recorded but in the dataset maximum 2 of them are used). Histograms on results distributions, mould and machine usage and other additional information are then displayed. Histograms are plotted both for unique and complete wheels. They are used to ensure that the two sets behave similarly so selecting unique wheels doesn't introduce some biases in the prediction.
Temporal series plots are instead exploited to reveal new issues on data measurements: each unusual behavior is discussed with the group experts in order to understand whether it is allowed or it is due to some errors. In the latter case an

update of filtering rules may be needed. These plots have also been used to search for a visual rule that could discriminate between OK and NOK wheels but, as Figure 2.4 shows, sometimes it is easy while some others it is not so there seems not to exist such a graphical rule.

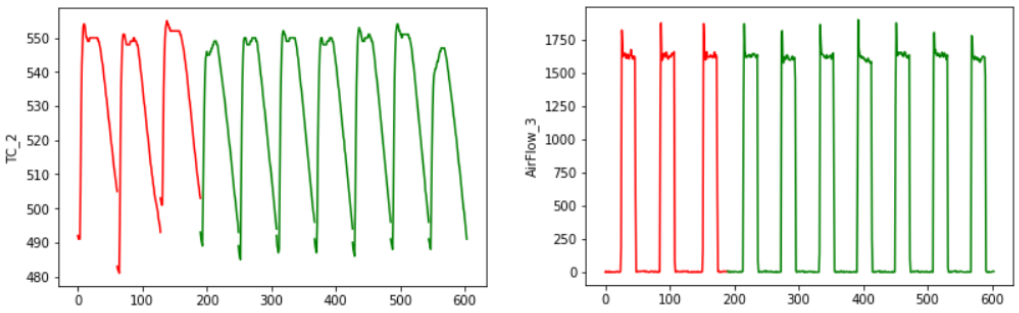

Figure 2.4: Plots of thermocouple (left) and airflow (right) measurements on which NOK wheels are marked in red and OK ones in green. Sometimes it is easy to tell if a wheel is discarded (last two NOK wheels shows a strange TC distribution) but some others it is not. Parameters such as airflow don't seem helpful in visual discrimination since their trend looks the same for each wheel.

3
Methods for prediction

This chapter is dedicated to the introduction of the statistical learning framework and of the techniques used for prediction. This chapter is heavily inspired by [4].

3.1 Stastical framework and key concepts

In order to explain the prediction methods used afterwards, it is useful to introduce a statistical framework that describes the data, the classifiers and their results. The mathematical notation adopted represents scalars with lower case letters (eg. y), (column) vectors with bold letters (eg. $\mathbf{x}$), uppercase letters represent sets or matrices (eg. set S) while x_i has been used in order to refer to the i-th element of vector $\mathbf{x}$*.

Data is represented as a vector of features $\mathbf{x}$ coming from a *domain set* $\mathcal{X}$ of object that we wish to label. Labels y come from a *label set* $\mathcal{Y}$ of all possible labels associated to the data. The *training set* is the set of labeled domain points represented as pairs $S = ((x_1, y_1), ..., (x_m, y_m))$ in $\mathcal{X} \times \mathcal{Y}$. The learner takes S as an input and it outputs a function called *prediction rule* $h : \mathcal{X} \to \mathcal{Y}$. This rule is also referred as *predictor* or *classifier* and it associates a label for each input

*By the given definitions, x_i represents a scalar so it is the $i-$th element of vector $\mathbf{x}$ while $\mathbf{x}_i$ represents a vector so it is the $i-$th one of a collection of vectors.

point $\widetilde{\mathbf{x}} \in \mathcal{X}$, whether there exists a $\widetilde{y} \in \mathcal{Y}$ such that $(\widetilde{\mathbf{x}}, \widetilde{y}) \in S$ or it is a new domain point.
It is possible to model the process as follows: an instance $\mathbf{x} \in \mathcal{X}$ is generated following a probability distribution $\mathcal{D}$ over $\mathcal{X}$ unknown to the learner[†] and its value $y \in \mathcal{Y}$ is given by a "correct"[‡] labeling function $f : \mathcal{X} \to \mathcal{Y}$ such that $y_i = f(\mathbf{x_i})$ for every i. Each dataset tuple $(\mathbf{x_i}, y_i)$ is then composed by sampling a point $\mathbf{x_i} \in \mathcal{X}$ according to $\mathcal{D}$ and by assigning the label $y_i = f(\mathbf{x_i})$.
The goal of h is then to emulate the unknown labeling function f as closely a possible. In order to determine each predictor's performances it has been defined an error metric as the probability of predicting a wrong label on a random dataset sampled from $\mathcal{D}$. The loss of predictor h with respect to probability distribution $\mathcal{D}$ and labeling function f is then described as:

$$L_{\mathcal{D},f}(h) \stackrel{def}{=} P_{\mathbf{x}\sim\mathcal{D}}[h(\mathbf{x}) \neq f(\mathbf{x})] \tag{3.1}$$

$L_{\mathcal{D},f}(h)$ is called *generalization error*, *generalization loss* or *true error* of h. If f is obvious then $L_{\mathcal{D}}(h)$ is used. Its calculation is impossible since the learner doesn't know both $\mathcal{D}$ and f so it is estimated numerically via the *empirical risk* (or *empirical error*) $L_S(h)$ defined as the number of wrongly predicted elements in the set divided by its size:

$$L_S(h) \stackrel{def}{=} \frac{|(\mathbf{x_i}, y_i) \in S : h(\mathbf{x_i}) \neq y_i|}{|S|} \tag{3.2}$$

By adapting these concepts to the wheel prediction model:

- The *domain set* $\mathcal{X}$ is equal to $\mathbb{R}^d$ where d is the dimension of the vector representing a wheel. A wheel is then characterized by a vector $\mathbf{x} \in \mathcal{X}$ of d real numbers.

- *Label set* $\mathcal{Y}$ is equal to $\{1, -1\}$ where 1 represents an OK wheel and -1 stands for a NOK wheel.

- The dataset S is then made of tuples $(\mathbf{x_i}, 1)$ if the i-th wheel is OK, $(\mathbf{x_i}, -1)$

[†]Otherwise it may be possible to "create" new input data points by just sampling some elements from that distribution.

[‡]This constraint may be relaxed since errors and noises affect the predicted value as well.

otherwise

- As a measure to evaluate the performances of the predictor two methods have been used. The first one corresponds to the *empirical risk* $L_S(h)$ while the second one measures the degree of certainty in wrong predictions. The predicting methods used return a probability associated to each label $P(1)$ and $P(-1) = 1 - P(1)$ for then assigning a result to each wheel by just selecting the maximum. By supposing to have a misclassified wheel labeled with the value y_{wrong}, the value $|0.5 - P(y_{wrong})|$ measures how "bad" is a wrong prediction, namely how much the predictor was confident in assigning the wrong label to the wheel. This loss measurements takes values in $(0, 0.5]$ where values close to 0 mean that the probability of selecting the two labels were similar (so it may be possible to fix the prediction by hard tuning the parameters) while values close to 0.5 mean that the classifier was quite confident in labeling the wheel with the erroneous tag.

Since the empirical risk is a snapshot of the generalization error on the sampled dataset S, one may think of a procedure (*Empirical Risk Minimization* or *ERM*) that searches for a predictor that minimizes the empirical error. This procedure however may fail miserably. Think about having a uniform distribution probability $\mathcal{D}$ over a given interval and a labeling function f such that $P(-1) = 0.5$ and $P(1) = 0.5$. By considering the following predictor

$$h_S(\mathbf{x}) = \begin{cases} y_i & \text{if } \exists\ (\mathbf{x_i}, y_i) \in S \text{ such that } \mathbf{x_i} = \mathbf{x} \\ -1 & \text{otherwise} \end{cases} \tag{3.3}$$

it is obvious that $L_S(h_S) = 0$ while $L_{\mathcal{D},f}(h_S) = 1/2$. This kind of classifier is then perfect on the dataset while having poor performances on new and unseen data. This behavior is called *overfitting* and it occurs when the classifier isn't able to generalize its decision rule to new data.
In order to understand how to avoid overfitting it is important to understand how the generalization error is composed. By considering a finite set of hypotheses (or *hypotheses class*) $\mathcal{H}$ that describes the characteristics of the predictors found by a specific algorithm, it is possible to decompose the generalization of $h_S \in \mathcal{H}$

as a sum of two different entities.

$$L_D(h_S) = \epsilon_{app} + \epsilon_{est} \text{ where: } \epsilon_{app} = \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \text{ and } \epsilon_{est} = L_{\mathcal{D}}(h_S) - \epsilon_{app} \quad (3.4)$$

The first value is the *approximation error* and it represents the error of the best predictor in $\mathcal{H}$. Its value does not depend on the number of elements on the dataset but rather from the hypothesis class chosen since it is possible to select a more accurate classifier by choosing a more complex hypothesis class. The second parameter is the *estimation error* and it represents the distance of the current predictor from the optimum one. This value may be decreased by increasing the dataset used to train the model or by selecting an easier hypothesis class with less parameters to be tuned. This analysis highlights the *bias-complexity tradeoff* for which selecting a rich $\mathcal{H}$ might lead to overfitting (because of having an high ϵ_{est}) while restricting it may not be suitable to cover the complexity of the data leading then to underfitting.
In order to understand predictor's performances, empirical risk is compared to the performances of some naive descriptors. These classifiers label the elements randomly or uniformly using only one label, in both cases without looking at the features therefore failing at generalizing its characteristics. In the case of random label assignment the predictor is wrong on average

$$L_S(h_{random}) = \frac{\text{number of classes} - 1}{\text{number of classes}} = \frac{|\mathcal{Y}| - 1}{|\mathcal{Y}|}$$

times while in the latter case error is

$$L_S(h_{uniform}) = \frac{\text{total number of elements - number of elements in the most popolar class}}{\text{total number of elements}}$$

By applying this reasoning to wheel casting problem, the first predictor is wrong 50% of the times while the second

$$L_S(h_{uniform}) = \frac{\text{number of NOK wheels}}{\text{total number of wheels analyzed}} \leq 5\%$$

since OK wheels are more than NOK ones[§]. The second descriptor has better

[§]The value of this approximation is found empirically and may change by considering different datasets from the one given.

performances than the first and it is then used as a comparison for the classifiers found using the specific algorithms presented in the following sections.

For the sake of completeness, it is common to distinguish between linear and non linear predictors. By defining the class of affine functions of parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ as

$$L_d = \left\{ h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \sum_{i=1}^{d} w_i x_i + b : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \right\} \tag{3.5}$$

or using the compact notation

$$L_d = \left\{ h_{\mathbf{w}'}(\mathbf{x}) = \langle \mathbf{w}', \mathbf{x}' \rangle = \sum_{i=1}^{d+1} w_i' x_i' : \mathbf{w}' = (\mathbf{w}, b), \mathbf{x}' = (\mathbf{x}, 1), \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \right\} \tag{3.6}$$

it is possible to describe the different hypothesis classes of linear predictors as a composition of a function $\phi : \mathbb{R} \to \mathcal{Y}$ on L_d. Common choices of ϕ are the sign function for binary classification purposes and the identity function for regression ones. Non linear predictors, instead, they use more advanced calculations to divide the data into different classes. Given their simplicity, linear predictors may be learnt efficiently and they are easy to interpret while non linear ones are more complex but may fit the data more accurately.

When trying to evaluate predictor's performances, it is of key importance to use a test set T of fresh data not used to create the classifier in order to have the most unbiased estimation possible. Sometimes it is also necessary to choose between models with different parameter settings and this is usually done by selecting the best method over a validation set V. The standard procedure is then to split the input data into 3 disjoint sets S, V, T for training, validation and testing purposes respectively[¶]. The different models $h_1, ..., h_n$ are trained on the training set S: the best performing one on V h_{best} is then selected as best model and its error rate is finally calculated over test set T.
Another technique to choose the best model is *k-fold cross validation* procedure that divides the training set in k subsets (folds) and uses each fold to evaluate the

[¶]In case of having only one model, the input dataset is splitted in 2: training set S and test set T.

performances of the predictor trained on the other k-1 folds. By averaging the value over all subsets, only the overall best performing parameters are kept, the predictor is trained on the complete training set and its performances are then evaluated on the test set.

3.2 Logistic Regression

Logistic regression applies a sigmoid function

$$\phi_{sig}(z) = \frac{1}{1+e^{-z}} \tag{3.7}$$

to the result of an affine function L_d. Using compact notation for affine spaces, logistic regression hypothesis class is therefore

$$H_{sig} = \phi_{sig} \circ L_d = \{\mathbf{x} \mapsto \phi_{sig}(\langle \mathbf{w'}, \mathbf{x'} \rangle) : \mathbf{w'} = (\mathbf{w}, b), \mathbf{x'} = (\mathbf{x}, 1), \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{3.8}$$

that maps vectors of d real values into a value in (0,1). The sigmoid function $\phi_{sig}(z)$ is an S-shaped relation that takes values close to 1 if z is very large and it takes values close to 0 if z is very small. These properties justify its use in binary classification task to predict the probability of belonging to a certain class: if the return value is higher than 0.5 the corresponding input is associated to class number 1 otherwise it is classified as belonging to the other class. A nice property of this descriptor is the possibility to value the estimation confidence: if the probability is close to 0.5 then the predictor is not sure about which class to assign while if it is close to 0 or 1, the classifier is quite confident of its result.
By supposing the label set to be $\mathcal{Y} = \{-1, 1\}$, a suitable error metric for logistic regression is a function that takes big values if the real label is $y = 1$ and the predicted value is $h(x) = -1$ or vice versa. A function like

$$l_A(h_{\mathbf{w}}, (\mathbf{x}, y)) = \frac{1}{1+e^{y\langle \mathbf{x}, \mathbf{w} \rangle}} \tag{3.9}$$

shows the desired behavior since it takes big values if y and $\langle \mathbf{x}, \mathbf{w} \rangle$ have different sign and low ones otherwise. Since the sign of the latter term is associated to the

predictor's output[‖], the given function takes big values if the real label y and the predicted value $h(x)$ differ. Any other function behaving like l_A is a reasonable loss function: a particular example is

$$l_B(h_{\mathbf{w}}, (\mathbf{x}, y)) = \log\left(1 + e^{-y\langle \mathbf{x}, \mathbf{w}\rangle}\right) \tag{3.10}$$

that in addition shows the convexity property with respect to $\mathbf{w}$.

The predictor weights are found by using an optimization algorithm that minimizes an error function. Empirical risk minimization is applied and it searches for the best fitting set of weights $\mathbf{w_{best}}$ on the training set $S = ((\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m))$ such that

$$\mathbf{w_{best}} \in \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 + e^{-y_i\langle \mathbf{x_i}, \mathbf{w}\rangle}\right) \tag{3.11}$$

Given the convexity of l_B this can be solved efficiently using standard optimization techniques.

3.2.1 Regularization

It may happen that logistic regression's predictors overfit the data and in those cases it is necessary to reduce hypothesis' class size in order to lower the estimation error ϵ_{est}. A standard technique to achieve this result is called regularization and it applies a penalty to complex descriptors. More specifically, this approach modifies the ERM rule into a *Regularized Loss Minimization* (*RLM*) that jointly minimizes the sum of the empirical risk and of a regularization function. This latter function may be seen as a complexity measure of the hypotheses or as a stabilizer of the learning algorithm.
RML searches for the set of weights $\mathbf{w_{best}}$ such that

$$\mathbf{w_{best}} \in \underset{\mathbf{w}}{\operatorname{argmin}}(L_S(\mathbf{w}) + R(\mathbf{w})) \tag{3.12}$$

[‖]By approaching it empirically if $\langle \mathbf{x},\mathbf{w}\rangle$ is positive then $e^{-\langle \mathbf{w},\mathbf{x}\rangle} < 1$ so $\frac{1}{1+e^{-\langle \mathbf{w},\mathbf{x}\rangle}} > 0.5$ then the assigned label is $h(x) = 1$. In the case of $\langle \mathbf{x},\mathbf{w}\rangle$ being negative then $e^{-\langle \mathbf{w},\mathbf{x}\rangle} > 1$ so $\frac{1}{1+e^{-\langle \mathbf{w},\mathbf{x}\rangle}} < 0.5$ then the assigned label is $h(x) = -1$.

where the regularization is the function $R : \mathbb{R}^d \to \mathbb{R}$. Regularized loss minimization is then a way to find a trade-off between low empirical risk and low complexity. Plenty of regularizers exist: the ones used in this thesis are L1, L2 and elastic net.
L1 regularizer uses the L1 norm as a measure of complexity resulting in $R(\mathbf{w}) = \lambda||\mathbf{w}||_1 = \lambda \sum_{i=0}^{d} |w_i|$. The associated RML will then find the set of weights $\mathbf{w_{best}}$ such that

$$\mathbf{w_{best}} \in \underset{\mathbf{w}}{\operatorname{argmin}}(L_S(\mathbf{w}) + \lambda||\mathbf{w}||_1) \tag{3.13}$$

In this formula λ controls the trade-off between complexity and accuracy on the training set: the higher its value the easier the predictor, while if it takes low values complex predictors with lower empirical risk are preferred.
L2 regularizer uses instead the L2 norm as a measure of complexity resulting in $R(\mathbf{w}) = \lambda||\mathbf{w}||_2 = \lambda \sum_{i=0}^{d} w_i^2$. The associated RLM will then find the set of weights $\mathbf{w_{best}}$ such that

$$\mathbf{w_{best}} \in \underset{\mathbf{w}}{\operatorname{argmin}}(L_S(\mathbf{w}) + \lambda||\mathbf{w}||_2) \tag{3.14}$$

As in the L1 case, λ controls the accuracy-complexity trade-off.
The two regularizers despite being structurally similar, they find different parameter settings. The most noticeable difference is on parameters shrinking on λ changes. As depicted in Figure 3.1, as λ grows using L1 regularization only a few parameters differ from zero while using L2 lot of them have small but different from zero values [5] [1]. This implies that classifiers found using L1 are easier to understand since they use less parameters for the prediction.
Elastic net uses a combination of L1 and L2 losses $R(\mathbf{w}) = \lambda_1||\mathbf{w}|| + \lambda_2||\mathbf{w}||_2$ in order to exploit both their advantages: having good explicative models and ease them setting some parameters at 0. The associated RML will then find the set of weights $\mathbf{w_{best}}$ such that

$$\mathbf{w_{best}} \in \underset{\mathbf{w}}{\operatorname{argmin}}(L_S(\mathbf{w}) + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2) \tag{3.15}$$

A naive approach may suggest to apply a two-step algorithm that calculates applies a L2 regularizer and them adjusts the weights using a L1 rule but that cause a double shrinkage of the parameters that seems not to be helpful to reduce

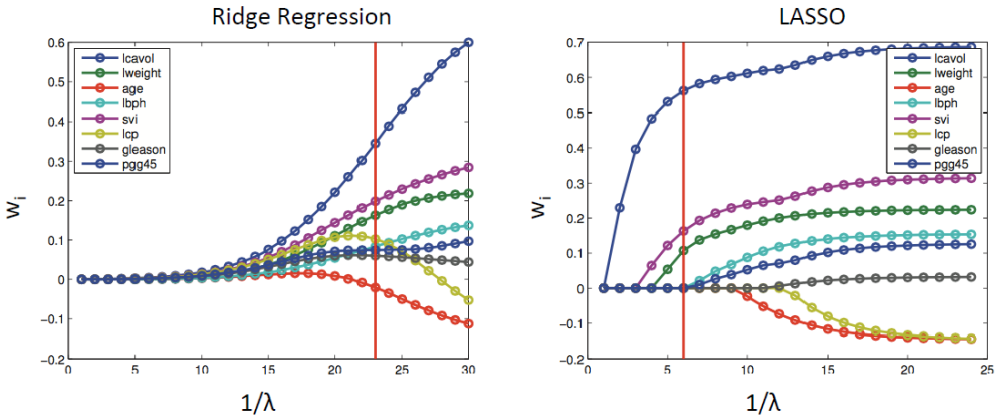

Figure 3.1: Parameters shrinkage by changing λ using L2 and L1 penalties. [1]

data variability. The correct way to implement the rule is to modify the weights using an L2 rule, compensate L2 shrinkage using a normalization term and then calculate the L1 regularization as suggested in [6].

3.3 SVM

Let suppose the label set to be $\mathcal{Y} = \{-1, 1\}$ and training set $S = ((\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m))$ to be linearly separable so there exists a line described by a set of weights $\mathbf{w}$ and a bias term b that perfectly divides the points labeled with $y = -1$ from those with label $y = 1$. Mathematically speaking that means that $\forall(\mathbf{x_i}, y_i) \in S : y_i = sign(\langle \mathbf{w}, \mathbf{x_i} \rangle + b)$ or alternatively** that $\forall(\mathbf{x_i}, y_i) \in S : y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 0$.
For any separable training set there may exist many couples $(\mathbf{w}, b)$ that separate the dataset correctly so a question arises: which one of them is the best? By looking at Figure 3.2 both lines separates the data but one may prefer the righter one since it is more distant its closest points. By defining as *margin* this distance, it is easy to notice how lines with an higher margin divide the data in a better way since they lower classification errors due to noises. By normalizing the weight vector $\mathbf{w}$, one may define the minimum distance of a dataset of points from the line as $\min_{i \in [0,m)} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b|$. *Hard support vector machine* (*Hard-SVM*) [7] [8] searches for the line with the biggest margin from the data that correctly

**This can be proved empirically since y_i and $(\langle \mathbf{w}, \mathbf{x_i} \rangle + b)$ must be of the same sign in order for their product to be positive.

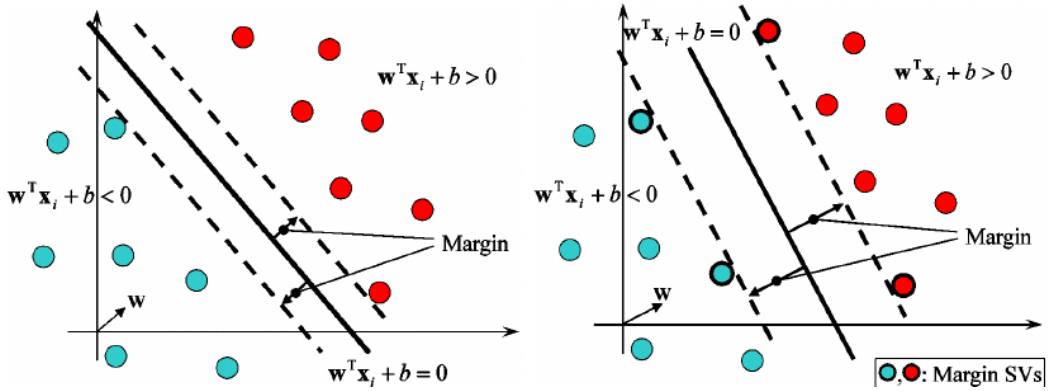

Figure 3.2: Two lines dividing the data. The righter one is preferred since it has an higher margin [2].

separates the two classes so a couple weight-bias such that

$$\underset{(\mathbf{w},b):||\mathbf{w}||=1}{\operatorname{argmax}} \min_{i\in[0,m)} |\langle \mathbf{w}, \mathbf{x_i}\rangle + b| \text{ such that } \forall(\mathbf{x_i}, y_i) \in S, y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) > 0 \quad (3.16)$$

or alternatively

$$\underset{(\mathbf{w},b):||\mathbf{w}||=1}{\operatorname{argmax}} \min_{i\in[0,m)} y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \quad (3.17)$$

Hard-SVM calculation can be efficiently solved as a quadratic optimization problem that searches for the minimum of a convex quadratic function having linear inequalities as constraints. A dual representation of the problem that uses inner products between vectors is described in Appendix B.

Support vector machine takes its name from support vectors that are those elements at minimum distance from the separation line, namely those that define the margin width. It has been demonstrated that the weight vector $\mathbf{w}$ is a linear combination of support vectors.

Despite being a powerful method, Hard-SVM requires the dataset to be linearly separable which is a strong condition that may not be met in practice. A relaxation of this constraint has been studied in *Soft-SVM* that introduces some slack variables $\xi_1, ..., \xi_m \geq 0$ that measure by how much the separation constraint is violated. The separation constraint becomes then

$$y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) > 1 - \xi_i \quad (3.18)$$

and the optimization problem can be rewritten as

$$\min_{\mathbf{w},b,\boldsymbol{\xi}}(\lambda||\mathbf{w}||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i) \tag{3.19}$$

The algorithm minimizes jointly the norm of $\mathbf{w}$[††] and the average of the slack variables in order to find the separation line with the highest margin that misses the fewest points. This trade-off is controlled by a parameter $\lambda > 0$: if λ is very big then the SVM will try to enlarge the margin while if it is close to 0 the algorithm will select the line that misses the lowest number of elements.

3.3.1 Kernels

SVM is a powerful algorithm but it is still limited to linear models. An extension to non-linear patterns may be to transform the training set S into another set $S' = ((\psi(\mathbf{x_1}), y_1), ..., (\psi(\mathbf{x_m}), y_m))$ using a non-linear function $\psi()$ in order to learn a predictor $\hat{h}$. Any new instance $\mathbf{x}$ will be then transformed using $\psi()$ and finally given to $\hat{h}$ for the classification task.
By considering the dual formulation introduced in Appendix B one may notice that transforming the data using ϕ is not mandatory since the calculation is based on inner products. The knowledge of the *kernel function* $K_\psi(\mathbf{x},\mathbf{x'}) = \langle\psi(\mathbf{x}), \psi(\mathbf{x'})\rangle$ is then the only constraint to train the SVM and that may allow some speedups in calculation[‡‡]. It is convenient to think the kernel function that uses inner products to evaluate the similarity between instances. A final remark is about incorporating prior knowledge in kernel choices. If one thinks that the positive samples can be distinguished by a circle then he may define ϕ as a measure of distance from its center, for example.
A variety of kernels are shown in the literature but in the thesis only four of them have been exploited:

- linear kernel $K_{lin}(\mathbf{x}, \mathbf{x'}) = \langle\mathbf{x}, \mathbf{x'}\rangle$ maps the inner product in $\mathbb{R}^d$ in itself. Basically this doesn't produce any transformation of the data and it is

[††]It is equivalent at maximizing the margin as shown in Appendix B.
[‡‡]In the literature this phenomena is called "kernel trick" since transforming the data and then calculating the inner products may take a certain amount time while calculating directly the value of the inner product may require a shorter one.

equivalent as using Hard/Soft-SVM as described in Section 3.3.

- polynomial kernel $K_{poly}(\mathbf{x}, \mathbf{x'}) = (\gamma\langle \mathbf{x}, \mathbf{x'}\rangle + r)^d$ implements a polynomial of degree d. In the thesis tested values are $d = \{2, 3, 4, 5, 6, 7\}$.
- radial basis function (RBF) kernel $K_{RBF}(\mathbf{x}, \mathbf{x'}) = e^{-\gamma||\mathbf{x},\mathbf{x'}||^2}$ implements a Gaussian kernel in which parameter $\gamma > 0$ equals the quantity $\frac{1}{2\sigma}$.
- sigmoid kernel $K_{sig}(\mathbf{x}, \mathbf{x'}) = \tanh(\gamma\langle \mathbf{x}, \mathbf{x'}\rangle + r)$ maps the points using a sigmoid-like function.

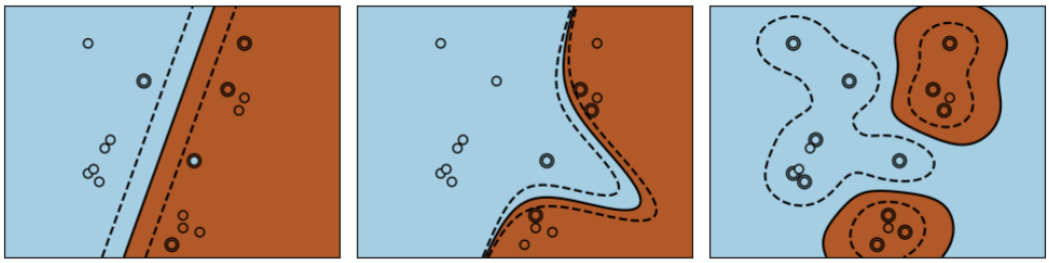

Figure 3.3: SVM using Linear kernel (left), degree 3 polynomial kernel (center), RBF kernel (right) on a data distribution. Support vectors are highlighted with bigger points [3].

4

Data classification results

This chapter is dedicated to the presentation of prediction results. Multiple tests have been implemented by varying parameters number, data quantity and optimization solvers. This chapter will use the notation of Appendix C to refer specific datasets by concatenating of the project number, a letter {s,m,c} indicating dataset size (respectively s=small, m=medium, c=complete) and, in the case of the extended dataset, an additional "e". Full result log is available at Appendix D. Reading both appendixes C and D is suggested in order to have a complete understanding of datasets characteristics and prediction evaluation measurements.

4.1 Influence of data quantity in prediction

As described in Chapter 3, the generalization error of a given predictor depends on data quantity so in this section it has been studied its influence on prediction accuracies. To do so, classification results have been compared on all datasets of wheel number 3701 both with standard and extended parameter set.
As shown in Table 4.1 by increasing dataset size also prediction results on the test set improve. By using a small dataset chances of overfitting are higher (both logistic regression with L1 and SVM with RBF kernel show very good performances on training set but lack on generalizing results on test set) while using larger

datasets lowers the gap between training and test results, increasing consistency therefore returning more stable classifiers. It is interesting to notice how SVM using RBF kernel needs lot of data to be effective as it overfits the small dataset and it predicts nicely the extended one.

Technique	**Training**		**Test**	
	IMP	**CON**	**IMP**	**CON**
Standard solvers				
(s) Logistic regression + L1	+68.10%	94/100	+10.10%	61/100
(m) Logistic regression + L1	+34.67%	89/100	+9.17%	68/100
(c) Logistic regression + L1	+32.10%	97/100	+14.14%	95/100
(s) SVM + RBF kernel	+88.80%	100/100	+3.90%	37/100
(m) SVM + RBF kernel	+62.28%	100/100	+6.61%	55/100
(c) SVM + RBF kernel	+40.21%	100/100	+23.76%	97/100
(s) SVM + poly kernel (deg.2)	+60.30%	96/100	+21.00%	91/100
(m) SVM + poly kernel (deg.2)	+40.11%	89/100	+10.06%	73/100
(c) SVM + poly kernel (deg.2)	+26.17%	86/100	+10.21%	83/100
Stochastic solvers				
(s) Logistic regression (SAGA)	+51.20%	100/100	+25.50%	93/100
(m) Logistic regression (SAGA)	+34.61%	100/100	+20.78%	94/100
(c) Logistic regression (SAGA)	+28.69%	100/100	+22.07%	99/100
(s) Logistic regression + L2 (SAGA)	+54.60%	100/100	+30.60%	97/100
(m) Logistic regression + L2 (SAGA)	+34.94%	99/100	+18.17%	96/100
(c) Logistic regression + L2 (SAGA)	+29.90%	100/100	+18.45%	97/100

Table 4.1: Training and test result comparison for five techniques on small (s), medium (m) and complete (c) dataset for project number 3701. IMP measures the percentage increment in performances of a predictor with respect to the naive classifier while CON counts the consistency of the result namely how many times it overperformed the naive predictor.

An opposite behavior appears when considering SVM with polynomial kernels showing nice results when applied on small dataset that are not replicated on medium and complete datasets. This may be due to small dataset's characteristics that may not emulate closely the probability distribution but resembles a

particular one that is easily classified with a degree 2 polynomial. By sampling more data, however, the dataset may lose its easy-to-predict shape resulting in another one more complex and with lower predicting performances. This is supported by the fact that on average classification results on small dataset are better than on medium size one. Despite that, using the complete dataset is still preferable since the greater generalization ability (shown as an improvement in consistency of results) and the higher discriminative power between effective methods (whose performances are improved with respect to small datasets) and weak ones (whose performances are worsened).
Stochastic solvers behave similarly to SVM with polynomial kernels but maintaining higher performances. Extended parameters set shows the same behavior having on average smaller but more consistent improvements.

4.2 Influence of standard and extended set in predictions

Adding parameters may enhance classifiers' performances by incrementing their generalization power but it also increases class complexity therefore it may lead to overfitting. This test had been performed to investigate this relationship by comparing results over standard and extended parameter sets for wheel number 3701 and 4273. Whenever possible, different dataset sizes have been used to study the trade-off between parameters number and data quantity.
Extended parameter affects nicely prediction in small datasets by increasing performances of 4/5% on average while losing effectiveness as the dataset size grows. On 4273 dataset, extended parameter set shows a bad impact on classifiers lowering their accuracy up to 7%. This behavior may be due to noisy additional observations, therefore more data may be required to achieve a good prediction. Consistency of predictions generally follows the same trend as the improvements with respect to the naive classifier. Table 4.2 contains a useful snapshot of two predicting methods on the different datasets: data from stochastic solvers have been omitted since it follows the described trends.

Technique	Standard		Extended	
	IMP	CON	IMP	CON
(3701s) Logistic regression + L2	+20.30%	85/100	+25.00%	87/100
(3701s) SVM + RBF kernel	+3.90%	37/100	+5.50%	33/100
(3701m) Logistic regression + L2	+16.78%	87/100	+15.67%	91/100
(3701m) SVM + RBF kernel	+6.61%	55/100	+10.00%	61/100
(3701c) Logistic regression + L2	+15.03%	97/100	+15.83%	98/100
(3701c) SVM + RBF kernel	+23.76%	97/100	+22.62%	97/100
(4273) Logistic regression + L2	+22.86%	100/100	+22.14%	98/100
(4273) SVM + RBF kernel	+25.86%	96/100	+21.89%	97/100

Table 4.2: Test result comparison for standard and extended parameters set on small (s), medium (m) and complete (c) dataset for project number 3701 and on dataset for project 4273. IMP measures the percentage increment in performances of a predictor with respect to the naive classifier while CON counts the consistency of the result namely how many times it overperformed the naive predictor.

4.3 Influence of solvers in computational time

Computational time increases with data quantity and it might be an issue for performing tests given the big amount of repetition for each method. It has then been decided to study the influence of different solvers of scikit-learn in order to find the best one considering both efficiency and performances. This test had been performed on all 3701 datasets with standard parameters set but the results apply to the extended one as well.
The following solvers use coordinate descend (liblinear), Newton's method (newton-cg), limited memory Broyden-Fletcher-Goldfarb-Shanno Algorithm (lbfgs), stochastic average gradient (SAG), SAGA. Liblinear is for L1 penalization only, newtoncg, lbfgs and SAG are for L2 only while SAGA works with all penalization types. These two methods try to overperform stochastic gradient descend (SGD) by implementing

$$x^{k+1} = x^k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} y_i^k \quad \text{where } y_i^k = \begin{cases} f_i'(x^k) & \text{if } i = i_k \\ y_i^{k-1} & \text{otherwise} \end{cases} \tag{4.1}$$

for SAG [9] and

$$x^{k+1} = x^k - \gamma \left[f'_j(x^k) - f'_j(x^{k-1}) + \frac{1}{n} \sum_{i=1}^{n} f'_i(x^{k-1}) \right] \tag{4.2}$$

for SAGA* [10]. They rely heavily on a tollerance parameter that sets the accuracy of the result and on a maximum iteration number that prevents the algorithm to loop. If this value is reached, then the algorithm returns the best solution found. Table 4.3 contains time comparison of 5 different methods using standard solvers (liblinear or lbfgs) and stochastic ones such as SAG/SAGA. It is easy to see that, differently from every other method that has a maximum variation of 6 times between small and complete dataset runtime, logistic regression with an L1 regularizer solved with liblinear solver takes much longer as the dataset size increase. A valid alternative with big datasets and L1 penalization may be to use SAGA solver that achieves even better performances (those will be discussed in Section 4.4) in much less time. The same speed advantage cannot be achieved by using SAG or SAGA with L2 regularization since lbfgs is already faster than them but prediction performances differ as well. Between the two stochastic methods, however, it is preferable to use SAG since it achieves similar performances in half of the time. However, it is important to point out that the maximum number of iterations have been reached in every run using SAG or SAGA solver therefore their running time to get an equivalent solution of standard solvers† may be higher.

4.4 Early stop as a form of regularization

Another analysis has been performed by stopping logistic regression algorithms with stochastic solvers after 100 iterations before they converge to an optimal solution with a given approximation of 0.0001. By doing so, the trained predictor may get the input data specifics but without overfitting them therefore being regularized.

*Both SAG and SAGA use a learning rate parameter (respectively α_k and γ) to control the convergence speed. The choice of the best learning rate is a very studied problem in the scientific community that has not been investigate in this work.

†Therefore selecting a predictor that is close to the best one by less than a given tolerance parameter.

Technique	Test set		Time
	IMP	CON	s/iter
(s) Logistic regression + L1 (liblinear)	+10.10%	61/100	148.48
(m) Logistic regression + L1 (liblinear)	+9.17%	68/100	1729.97
(c) Logistic regression + L1 (liblinear)	+14.14%	95/100	95348.28
(s) Logistic regression + L2 (lbfgs)	+20.30%	85/100	166.01
(m) Logistic regression + L2 (lbfgs)	+16.78%	87/100	363.63
(c) Logistic regression + L2 (lbfgs)	+15.03%	97/100	788.09
(s) Logistic regression + L1 (SAGA)	+27.20%	94/100	965.91
(m) Logistic regression + L1 (SAGA)	+17.17%	89/100	1774.50
(c) Logistic regression + L1 (SAGA)	+18.69%	98/100	5888.99
(s) Logistic regression + L2 (SAG)	+29.10%	95/100	579.51
(m) Logistic regression + L2 (SAG)	+16.89%	90/100	1033.32
(c) Logistic regression + L2 (SAG)	+18.83%	99/100	3057.72
(s) Logistic regression + L2 (SAGA)	+30.60%	97/100	1200.88
(m) Logistic regression + L2 (SAGA)	+18.17%	96/100	2182.96
(c) Logistic regression + L2 (SAGA)	+18.45%	97/100	6820.12

Table 4.3: Test result comparison for 5 different methods on small (s), medium (m) and complete (c) dataset of wheel number 3701. IMP measures the percentage increment in performances of a predictor with respect to the naive classifier while CON counts the consistency of the result namely how many times it overperformed the naive predictor.

By looking at Table 4.3 it is clear how SAG and SAGA outperform lbfgs and liblinear both on improvement and consistency. This behavior is more evident the less the data confirming the intuition of an early stop as a rough regularization. It may be possible having more data, however, that this trend continues until the point when tolerance based algorithms (lbfgs and liblinear) outperform tollerance-and-iteration based ones such as SAG and SAGA.

4.5 Errors and improvements distributions analysis

Errors and improvements distributions analysis have been performed as well exploiting two different plots:

- Improvement distribution histograms that show both the training and test improvements over the naive descriptor during each of the 100 iterations.

- Prediction probability's distance from 0.5 of misclassified elements that measures the degree of certainty in wrong predictions as described in Section 3.1. These values as well are represented using an histogram.

Improvements histograms can be classified in two classes. A first one of Gaussian-like distributed elements usually represents good predictor whose average improvement are positive in contrast to a second, shaped similarly to a negative exponential, that is usual of bad classifiers and it has a peak around 0. Figure 4.1 shows one example per class.

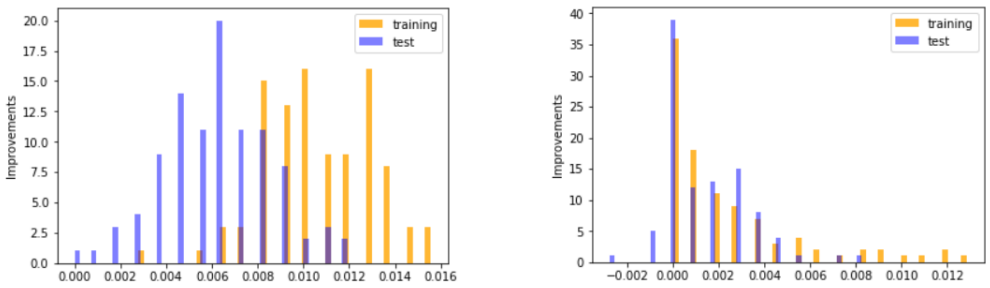

Figure 4.1: The two different behaviors of error rate improvement distributions: the left one (SVM + RBF kernel) represents a good predictor while the right one (SVM + polynomial kernel of degree 4) a bad performing one.

Histograms of distance from 0.5 of misclassified samples are instead quite similar despite being referred to good predictors (such as the one on the left image in Figure 4.2) or bad ones (central histogram). The only exception of this rule is represented by logistic regression trained with SAGA solver that shows a more spreaded and various trend as in right histogram of Figure 4.2. General distribution shows a peak around 0.5 and that means that lot of misclassified elements are wrongly predicted with an high certainty by the classifier. This behavior may suggest that current combination of models and parameters, despite having good performances, may not be enough to cover very accurately and completely dataset's variability.

4.6 Comparison between different wheels and methods results

Given the variety of wheels analyzed and data distributions, it is interesting to compare classifiers performances on all the datasets in order to detect significant

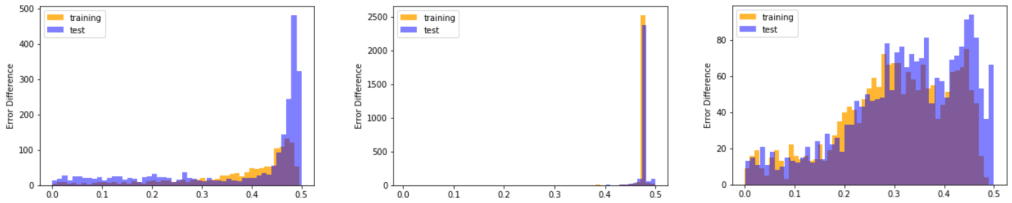

Figure 4.2: Histogram of distance from 0.5 of misclassified samples for SVM with RBF kernel (left), SVM with polynomial kernel of degree 5 (center) and logistic regression solved with SAGA (right).

behaviors.

The best performing methods on 3701 complete dataset are SVM with RBF kernel and logistic regression solved with SAGA solvers showing improvements around 22/23.7% with respect to naive predictor with a great confidence level. Better results are achieved when studying wheel 4273 with improvements peaks of 28% and several methods passing 20% all achieving nearly perfect confidence scores. Wheel number 4509 predictions, instead, are far away from these results achieving a 20% improvement peak on logistic regression with SAGA solver and a second best result of +12.56%. In addition, all 4509 classifiers cannot reach a good level of confidence stopping at a maximum level of 78/100 but with several values lower than 60/100. Table 4.4 helps results visualization with a quick resume of best methods per wheel type.

The reasons of these many differences may be multiple. At first data quantity plays an important role because some methods may overfit as depicted in Section 4.1 considering SVM with RBF kernel. Data distributions affect predictions as well, in particular wheel 4273 seems much easier to predict than 3701 since it achieves better performances despite having less data. NOK%, in particular, may play an important role given that the predictor may have difficulties in generalization if they train mainly on OK samples. Finally collected parameters may heavily affect classification. This could explain 4273 having better performances than 3701 since the former gathers information from 2 thermocouples measurements while the latter only from one. For the same reason, wheel 4509 may need some other sensor or parameter tuning that are now missing from the dataset leading to poor prediction performances.

Technique	Training		Test	
	IMP	CON	IMP	CON
Wheel number 3701				
(1) SVM + RBF kernel	+40.21%	100/100	+23.76%	97/100
(2) Logistic regression (SAGA)	+28.69%	100/100	+22.07%	99/100
(3) Logistic regression + L2 (SAG)	+33.17%	100/100	+18.83%	99/100
Wheel number 4273				
(1) Logistic regression + L1 (SAGA)	+38.93%	100/100	+28.68%	100/100
(2) SVM + poly kernel (deg.2)	+42.32%	98/100	+26.71%	99/100
(3) Logistic regression + L2 (SAG)	+39.68%	100/100	+25.43%	100/100
Wheel number 4509				
(1) Logistic regression (SAGA)	+41.00%	100/100	+20.44%	78/100
(2) Logistic regression + L1 (SAGA)	+43.70%	94/100	+12.56%	69/100
(3) Logistic regression + L2 (SAGA)	+42.30%	93/100	+11.44%	65/100

Table 4.4: Top 3 training and test results for each wheel number on its biggest dataset available. IMP measures the percentage increment in performances of a predictor with respect to the naive classifier while CON counts the consistency of the result namely how many times it overperformed the naive predictor.

5

Conclusions and future developments

This thesis was a feasibility study about the potential of machine learning algorithms applied to casting process data. The first stage of this work had been of reading and collecting data from a given casting machine (BP10). A dataset had then been created by removing outliers and data reading errors while only wheels with a specific project ID number had been taken into account. The next phase consisted in the creation of a vector representing a wheel by calculating statistical features from in-process measurements. Histogram and temporal series plots had been used for data visualization and analysis in order to find particular behaviors or untracked data writing errors. Finally, various models had been built and trained on the dataset with different outcomes.

Overall results of this preliminary test on quality prediction are good since for each wheel nearly all the classifiers behave better than the naive predictor and this is an indicator that current data contains information that can be exploited by machine learning algorithms.

In order to understand results properly, performance variability had been studied and, despite its complexity, it had been tackled on different aspects. Data quantity surely affects predictors quality since some methods need a good amount of data to generalize properly wheels characteristics and not overfit. NOK% or specific intrinsic wheel properties may influence effectiveness as well since the model is richer with examples of various type and not only well-casted wheels. If one

may be tempted to extend the parameter set, Section 4.2 shows that this should be done properly since at first they may help in prediction but they may not be affective as with higher data quantities. Bigger datasets, on the other side, take more time to be filtered and trained: runtime analysis shown that stochastic algorithms may help in speeding up computations but they are not always the fastest solution*. This class of algorithms surprisingly performs really well in all datasets despite lowering their effectiveness when dealing with high data quantities: this trend is expected to continue as more data is gathered. Finally error and improvement analysis had been performed suggesting how to discriminate well-fitting predictors from bad ones and that extra work is needed to increase prediction accuracy.

Performance increase may be achieved by trying other methods (one-class SVMs, random forests or, with an higher amount of data, neural networks), by improving used ones (for example implementing a non-trivial elastic net regularizer) or by extending the parameters set with the addition of thermocouples, that turned out from testing-time weight analysis to be within the most informative parameters, or with the creation of ad-hoc ones. Additional tests may involve predicting new wheels using old time series data (now shuffling is applied before splitting train and test sets so temporal correlations are lost) to check in-field classification effectiveness. Another option may be to generalize this method to new models, machines or moulds. An extension of data gathering systems may then be used to check performances variabilities and predictor generalization abilities. It may be worth trying to modify current methods by allowing them to use different parameters in prediction for example being able to exploit only information of one thermocouple if the others are not connected while using dynamically multiple values whenever two or more are attached. Lastly it would be nice to dig deeper into prediction understanding how parameters affect results in order to translate data expertise into human knowledge for productive process improvement.

*However this may not be a big problem if data is analyzed using batch processing techniques.

A

Data legend

This appendix is dedicated to list the different parameters recorded in the .csv files and how their information has been processed.

A.1 PRD data file

Here there is a list of all the parameters contained in PRD data files.

Field	**Description**
SerialNo	Serial number identifying a single wheel
ProjectID	Project ID. This is the wheel project number
MouldID	Mould ID. Represent the ID of a single mould for a specific project
ReturnCode	Return code from operator. See below.
shiftNumber	Shift number
MachineID	Machine ID
shiftStartTime	Shift start time
t_stamp	Row timestamp

Return code Table is the following. For the extent of the thesis, wheel casting is considered OK if return code is 20, it is not considered if return code is 30 otherwise it is NOK.

Id	Description
1	Restart after mould change
2	Restart after machine set
3	Restart after machine fault
4	Visible Cavity/Holes
5	Inclusions of dust/dirt
6	Other visible defects
7	Lack of alluminium in oven
8	Other defects
20	Wheel is OK and sent to the Conveyor
30	Wheel is manually removed from casting machine, thus is nor OK neither NOT OK.

A.2 PRM data file

Here there is a list of all the parameters contained in PRM data files. Each mould has up to 16 cooling circuits that may work in different modes.

Field	Description	Unit of Measure
SerialNo	Serial number identifying a single wheel. This is the key on all files for a single wheel	N/A
MachineID	Machine ID	N/A
Name	Machine Name	N/A
ProjectID	Project ID. This is the wheel project number	N/A
MouldID	Mould ID. Represent the id of a single mould fo a specific project	N/A
FinalPressure	Final pressure to reach for oven	mBar
Phase2Pressure	Switch pressure between phase 2 and phase 3. Phase 2 ramp is controlled by filling rate parameter.	mBar
FillingRate	Filling rate for phase 2	mBar/s
Oven_Temp	Oven temperature	℃
PressureTime	Pressure phase duration	sec
CoolingTime	Cooling phase duration	sec
CX_TY_AirFlow	For each Circuit number X and Time sequence Y this parameter set the quantity of air injected by the circuit	l/min
CX_TY_WorkTime	For each Circuit number X and Time sequence Y this parameter set how many seconds the circuit has to wait from the beginning of cycle to start working.	sec
CX_WorkMode	0 = Circuit disabled, 1-4 circuit working in Thermo mode (not used), 5 = Circuit working in timed mode	N/A
t_stamp	Record timestamp	N/A

A.3 PD data file

Here there is a list of all the parameters contained in PD data files.

Field	**Description**	**Unit of Measure**
SerialNo	Serial number identifying a single wheel. This is the key on all files for a single wheel	N/A
TC_1	Thermocouple temperature 1. Value 3276 means no TC connected.	℃
TC_2	Thermocouple temperature 2. Value 3276 means no TC connected.	℃
TC_3	Thermocouple temperature 3. Value 3276 means no TC connected.	℃
TC_4	Thermocouple temperature 4. Value 3276 means no TC connected.	℃
TC_5	Thermocouple temperature 5. Value 3276 means no TC connected.	℃
OvenAirFlow	Oven air inlet flow	l/min
Pressure	Oven overpressure	mBar
Temperature	Oven temperature	℃
AirFlow_X	Circuit X air flow	l/min
CCPressure_X	Cooling circuit pressure	mBar
OvenDoorOpen	1 if oven door is opened, 0 otherwise	N/A
EnvironmentTemp	Foundry temperature	℃
EnvironmentHygro	Relative environment humidity	N/A
t_stamp	Row time stamp	N/A

Standard set of process measurements contains all the parameters of the Table from thermocouple values to airflows while extended set contains also pressure, door opening and environment informations (except humidity). Door opening is

not treated as a process measurement (by calculating statistical descriptors over the time series) but it is used to calculate the number of wheels casted since last door opening.

A.4 XRAY data file

Here there is a list of all the parameters contained in XRAY data files.

Field	**Description**
SerialNR	Serial number identifying a single wheel. This is the key on all files for a single wheel. If 0 wheel has been loaded manually.
WheelCode	Project ID. This is the wheel project number
MouldID	Mould ID. Represent the id of a single mould for a specific project.
ND	Machine number. The number of machine that casted the wheel.
XRay	Xray machine. The number of machine that did the Xray inpection.
ErrorCodeXR	Error code from xray. See below.
t_stamp	Row time stamp

Return code Table is the following. For the extent of the thesis, wheel xray analysis is considered OK if return code is 0 otherwise it is NOK.

Id	**Description**
0	OK wheel
1	Cavity on spoke-channel link
2	Cavity on center spoke
3	Cavity on spoke-hub link
4	Cavity on hub
5	Bubbles on hub
6	Porosity/cavity on channel
7	Porosity inner edge
8	Crooked degating
9	Filter inclusion

B

Dual representation of SVM

This appendix introduces a dual representation of SVM optimization problem based on inner products that is fundamental for kernel use. This appendix is heavily inspired by [4].

As described in section 3.3, hard SVM tries to find the couple $(\mathbf{w}, b)$ that linearly separates the data with the highest margin by calculating

$$\underset{(\mathbf{w},b):||\mathbf{w}||=1}{\operatorname{argmax}} \min_{i\in[0,m]} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b| \text{ such that } \forall (\mathbf{x_i}, y_i) \in S, y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 0 \quad \text{(B.1)}$$

or given the fact that $y_i \in \mathcal{Y} = \{-1, 1\}$, its alternative form

$$\underset{(\mathbf{w},b):||\mathbf{w}||=1}{\operatorname{argmax}} \min_{i\in[0,m]} y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \quad \text{(B.2)}$$

Another equivalent formulation may be given as an optimization problem of a convex quadratic function with linear inequalities as constraints like

$$(\mathbf{w}_0, b_0) = \underset{(\mathbf{w},b)}{\operatorname{argmin}} ||\mathbf{w}||^2 \text{ such that } \forall (\mathbf{x_i}, y_i) \in S, y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \geq 1 \quad \text{(B.3)}$$

that forces the margin to be 1 but scales the measurement unit with the norm of $\mathbf{w}$. The following lemma demonstrates this statement.

Lemma B.0.1. *Equations (B.2) and (B.3) are two equivalent resolutions of hard-SVM problem*

Proof. Define $\gamma^* = \min_{i\in[0,m]} y_i(\langle \mathbf{w}^*, \mathbf{x_i}\rangle + b^*)$ as the minimum distance from the solution of equation (B.2) $(\mathbf{w}^*, b^*)$.
For all i then

$$y_i(\langle \mathbf{w}^*, \mathbf{x_i}\rangle + b^*) \geq \gamma^*$$

By dividing both members by γ^* it becomes

$$y_i(\langle \frac{\mathbf{w}^*}{\gamma^*}, \mathbf{x_i}\rangle + \frac{b^*}{\gamma^*}) \geq 1$$

so the vector $(\frac{\mathbf{w}^*}{\gamma^*}, \frac{b^*}{\gamma^*})$ satisfies (B.3) conditions therefore $||\mathbf{w}_0|| \leq ||\frac{\mathbf{w}^*}{\gamma^*}|| = \frac{1}{\gamma^*}$.
By defining $\hat{\mathbf{w}} = \frac{\mathbf{w}_0}{||\mathbf{w}_0||}$ and $\hat{b} = \frac{\mathbf{b}_0}{||\mathbf{w}_0||}$ it follows that

$$y_i(\langle \hat{\mathbf{w}}, \mathbf{x_i}\rangle + \hat{b}) = \frac{1}{||\mathbf{w}_0||} y_i(\langle \mathbf{w}_0, \mathbf{x_i}\rangle + b_0) \geq \frac{1}{||\mathbf{w}_0||} \geq \gamma^*$$

Since $||\hat{\mathbf{w}}|| = 1$ all conditions (B.2) are met, $(\hat{\mathbf{w}}, \hat{b})$ is an optimal solution. □

It is important to remember that a dual version that embeds the constraints in the maximization problem is available.
By considering the function

$$g(\mathbf{w}) = \max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} \sum_{i=1}^{m} \alpha_i(1 - y_i\langle \mathbf{w}, \mathbf{x_i}\rangle) = \begin{cases} 0 & \text{if } \forall i, y_i\langle \mathbf{w}, \mathbf{w_i}\rangle \geq 1 \\ +\infty & \text{otherwise} \end{cases} \tag{B.4}$$

it is possible to rewrite (B.3) as

$$\begin{aligned} &\min_{\mathbf{w}}(\frac{1}{2}||\mathbf{w}||^2 + g(\mathbf{w})) = \\ &= \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} (\frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} \alpha_i(1 - y_i\langle \mathbf{w}, \mathbf{x_i}\rangle)) \geq \\ &\geq \max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} \min_{\mathbf{w}} (\frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} \alpha_i(1 - y_i\langle \mathbf{w}, \mathbf{x_i}\rangle)) \end{aligned} \tag{B.5}$$

By fixing $\boldsymbol{\alpha}$, the optimization problem with respect to $\mathbf{w}$ is differentiable with optimum value at

$$\mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{B.6}$$

By substituting this result in the last equation of (B.5) it leads to

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq 0} (\frac{1}{2} || \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i ||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i \langle \sum_{j=1}^{m} \alpha_i y_i \mathbf{x}_j, \mathbf{x_i} \rangle)) \tag{B.7}$$

Rearranging the terms, the rule becomes

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq 0} (\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_j, \mathbf{x}_i \rangle) \tag{B.8}$$

It is important to notice how this formulation doesn't require the access to the elements $\mathbf{x}_i, \mathbf{x}_j$ but it only needs to know their inner product. This property is fundamental in the implementation of SVM with kernels.

C

Dataset presentation

This appendix contains information about the different datasets used for prediction. At first the overall dataset received is presented and then lot of small subset datasets used in prediction evaluation are described.

C.1 Overall dataset

Overall dataset is composed by 28 .csv files (7 per type described in Section 2.1) of 11895 complete wheels (11418 OK and 477 NOK) of which 11573 are unique (11108 OK and 465 NOK). Various moulds have been used of 14 different wheel types all casted by the same casting machine (BP10). 8819 of these wheels are collected with the extended parameter set of which 8577 are OK and 242 are NOK.
From this big dataset others are obtained by selecting only a specific project ID. They are described below: for the sake of simplicity their name is briefly summarized as the concatenation of the project number, a letter {s,m,c} indicating dataset size (respectively s=small, m=medium, c=complete) and, in the case of the extended parameter set, an additional ”e”.

C.2 Wheel number 3213

From the overall dataset only 17 refers to project ID number 3213. They are not enough to be analyzed by the prediction algorithm

C.3 Wheel number 3701

For wheel number 3701 it is possible to extract 6 different datasets:

- **3701s** counts 915 unique wheels, 895 OK (97.81%) and 20 NOK (2.19%). Vector dimension before non informative columns cleanup is 261 and after it is 93.
- **3701se** counts 915 unique wheels, 895 OK (97.81%) and 20 NOK (2.19%). Vector dimension before non informative columns cleanup is 347 and after it is 143.
- **3701m** counts 1642 unique wheels, 1606 OK (97.81%) and 36 NOK (2.19%). Vector dimension before non informative columns cleanup is 261 and after it is 93.
- **3701me** counts 1642 unique wheels, 1606 OK (97.81%) and 36 NOK (2.19%). Vector dimension before non informative columns cleanup is 347 and after it is 143.
- **3701c** counts 3824 unique wheels, 3766 OK (98.48%) and 58 NOK (1.52%). Vector dimension before non informative columns cleanup is 261 and after it is 137.
- **3701ce** counts 3824 unique wheels, 3766 OK (98.48%) and 58 NOK (1.52%). Vector dimension before non informative columns cleanup is 347 and after it is 200.

This wheel's moulds have only one thermocouple place that is related to TC_2.

C.4 Wheel number 4273

Wheel number 4273 had been predicted on two datasets:

- **4273c** counts 2183 unique wheels, 2127 OK (97.43%) and 56 NOK (2.57%). Vector dimension before non informative columns cleanup is 261 and after it is 101.
- **4273ce** counts 2183 unique wheels, 2127 OK (97.43%) and 56 NOK (2.57%). Vector dimension before non informative columns cleanup is 347 and after it is 175.

For this wheel both TC_1 and TC_2 are used to gather temperature information.

C.5 Wheel number 4509

Data related to wheel number 4509 has been collected using only standard set of parameters therefore only one dataset had been extracted. **4509c** counts 646 unique wheels, 627 OK (97.06%) and 19 NOK (2.94%). Vector dimension before non informative columns cleanup is 261 and after it is 87. Both TC_1 and TC_2 data have been collected.

D

Result logs

This chapter contains result logs of models performances on the different datasets described in C. The tests have been performed on 100 random shuffles of the data and 3 statistics have been evaluated. The error rate (ER) counts the mean number of misclassified elements, the improvement over naive predictor (IMP) studies predictor's real classification ability (IMP = ER - naive predictor ER) while consistency (CON) counts how many time the trained classifier overperformed the naive one. Tests have been performed on a system with a Intel Celeron E1200 CPU, 8 GB of RAM and an optical hard disk: this configuration surely is not a state of the art one but time values are collected in order to compare different methods, not for having an absolute idea of what is the runtime on modern computers.

D.1 Wheel number 3213

As described in the previous appendix, no training have been performed on wheel number 3213 since its data is not enough to ensure the training of a good model.

D.2 Wheel number 3701

Technique	Training ER	Training IMP	Training CON	Test ER	Test IMP	Test CON	Time s/iter
Standard solvers							
Logistic regression	0.0000	+0.0218 (+100.00%)	100/100	0.0254	-0.0035 (-16.20%)	25/100	2.73
Logistic regression + L1	0.0070	+0.0149 (+68.10%)	94/100	0.0197	+0.0022 (+10.10%)	61/100	148.48
Logistic regression + L2	0.0086	+0.0132 (+60.60%)	98/100	0.0174	+0.0044 (+20.30%)	85/100	166.01
SVM + linear kernel	0.0167	+0.0051 (+23.30%)	84/100	0.0186	+0.0033 (+15.10%)	78/100	3.18
SVM + poly kernel (deg.2)	0.0087	+0.0132 (+60.30%)	96/100	0.0173	+0.0046 (+21.00%)	91/100	3.30
SVM + poly kernel (deg.3)	0.0119	+0.0100 (+45.60%)	85/100	0.0195	+0.0023 (+10.70%)	68/100	3.87
SVM + poly kernel (deg.4)	0.0152	+0.0066 (+30.40%)	69/100	0.0200	+0.0019 (+8.50%)	52/100	4.57
SVM + poly kernel (deg.5)	0.0167	+0.0051 (+23.40%)	64/100	0.0205	+0.0014 (+6.20%)	49/100	6.94
SVM + poly kernel (deg.6)	0.0165	+0.0053 (+24.30%)	64/100	0.0202	+0.0017 (+7.70%)	48/100	6.41
SVM + poly kernel (deg.7)	0.0195	+0.0024 (+10.80%)	59/100	0.0206	+0.0013 (+5.80%)	41/100	10.12
SVM + RBF kernel	0.0024	+0.0194 (+88.80%)	100/100	0.0210	+0.0009 (+3.90%)	37/100	6.60
SVM + sigmoid kernel	0.0217	+0.0002 (+0.80%)	10/100	0.0218	+0.0001 (+0.50%)	5/100	2.90
Stochastic solvers							
Logistic regression (SAGA)	0.0107	+0.0112 (+51.20%)	100/100	0.0163	+0.0056 (+25.50%)	93/100	16.42
Logistic regression + L1 (SAGA)	0.0089	+0.0130 (+59.40%)	100/100	0.0159	+0.0060 (+27.20%)	94/100	965.91
Logistic regression + L2 (SAG)	0.0096	+0.0123 (+56.20%)	98/100	0.0155	+0.0064 (+29.10%)	95/100	579.51
Logistic regression + L2 (SAGA)	0.0099	+0.0119 (+54.60%)	100/100	0.0152	+0.0067 (+30.60%)	97/100	1200.88
Logistic regression + ElasticNet (SAGA)	0.0101	+0.0117 (+53.70%)	98/100	0.0155	+0.0064 (+29.20%)	95/100	1213.69

Table D.1: Results for wheel number 3701 on small dataset.

Technique	Training ER	Training IMP	Training CON	Test ER	Test IMP	Test CON	Time s/iter
Standard solvers							
Logistic regression	0.0000	+0.0218 (+100.00%)	100/100	0.0223	-0.0004 (-2.00%)	38/100	2.91
Logistic regression + L1	0.0058	+0.0160 (+73.50%)	97/100	0.0184	+0.0035 (+16.00%)	77/100	234.05
Logistic regression + L2	0.0074	+0.0144 (+65.90%)	99/100	0.0164	+0.0055 (+25.00%)	87/100	180.78
SVM + linear kernel	0.0164	+0.0054 (+24.90%)	87/100	0.0179	+0.0040 (+18.30%)	84/100	4.09
SVM + poly kernel (deg.2)	0.0084	+0.0134 (+61.50%)	97/100	0.0172	+0.0047 (+21.60%)	91/100	4.58
SVM + poly kernel (deg.3)	0.0110	+0.0108 (+49.40%)	95/100	0.0183	+0.0035 (+16.20%)	81/100	5.97
SVM + poly kernel (deg.4)	0.0140	+0.0078 (+35.90%)	91/100	0.0190	+0.0029 (+13.20%)	69/100	8.05
SVM + poly kernel (deg.5)	0.0157	+0.0062 (+28.20%)	79/100	0.0192	+0.0027 (+12.40%)	64/100	15.46
SVM + poly kernel (deg.6)	0.0178	+0.0041 (+18.60%)	84/100	0.0196	+0.0023 (+10.40%)	54/100	16.14
SVM + poly kernel (deg.7)	0.0185	+0.0034 (+15.40%)	76/100	0.0198	+0.0021 (+9.40%)	53/100	28.43
SVM + RBF kernel	0.0036	+0.0183 (+83.60%)	100/100	0.0207	+0.0012 (+5.50%)	33/100	8.47
SVM + sigmoid kernel	0.0217	+0.0001 (+0.50%)	5/100	0.0219	-0.0000 (-0.00%)	1/100	3.60
Stochastic solvers							
Logistic regression (SAGA)	0.0107	+0.0111 (+50.80%)	100/100	0.0159	+0.0060 (+27.30%)	90/100	24.79
Logistic regression + L1 (SAGA)	0.0090	+0.0129 (+59.00%)	100/100	0.0158	+0.0061 (+27.90%)	92/100	1419.77
Logistic regression + L2 (SAG)	0.0083	+0.0135 (+61.90%)	99/100	0.0153	+0.0065 (+29.90%)	97/100	737.46
Logistic regression + L2 (SAGA)	0.0100	+0.0118 (+54.20%)	100/100	0.0147	+0.0072 (+32.80%)	98/100	1681.69
Logistic regression + ElasticNet (SAGA)	0.0100	+0.0119 (+54.40%)	100/100	0.0147	+0.0071 (+32.60%)	100/100	1681.58

Table D.2: Results for wheel number 3701 using extended parameter set on small dataset.

Technique	Training			Test			Time
	ER	IMP	CON	ER	IMP	CON	s/iter
Standard solvers							
Logistic regression	0.0000	+0.0219 (+100.00%)	100/100	0.0373	-0.0154 (-70.17%)	0/100	9.57
Logistic regression + L1	0.0143	+0.0076 (+34.67%)	89/100	0.0199	+0.0020 (+9.17%)	68/100	1729.97
Logistic regression + L2	0.0144	+0.0076 (+34.50%)	95/100	0.0182	+0.0037 (+16.78%)	87/100	363.63
SVM + linear kernel	0.0200	+0.0020 (+9.00%)	74/100	0.0205	+0.0014 (+6.44%)	64/100	13.41
SVM + poly kernel (deg.2)	0.0131	+0.0088 (+40.11%)	89/100	0.0197	+0.0022 (+10.06%)	73/100	9.66
SVM + poly kernel (deg.3)	0.0162	+0.0058 (+26.28%)	88/100	0.0200	+0.0020 (+9.00%)	71/100	12.13
SVM + poly kernel (deg.4)	0.0196	+0.0023 (+10.44%)	40/100	0.0213	+0.0006 (+2.78%)	33/100	13.28
SVM + poly kernel (deg.5)	0.0202	+0.0017 (+7.67%)	60/100	0.0210	+0.0009 (+4.06%)	44/100	16.22
SVM + poly kernel (deg.6)	0.0211	+0.0008 (+3.61%)	39/100	0.0214	+0.0006 (+2.56%)	33/100	16.32
SVM + poly kernel (deg.7)	0.0208	+0.0011 (+5.17%)	63/100	0.0212	+0.0007 (+3.11%)	46/100	24.20
SVM + RBF kernel	0.0083	+0.0137 (+62.28%)	100/100	0.0205	+0.0014 (+6.61%)	55/100	17.93
SVM + sigmoid kernel	0.0218	+0.0001 (+0.50%)	5/100	0.0218	+0.0001 (+0.44%)	4/100	7.57
Stochastic solvers							
Logistic regression (SAGA)	0.0143	+0.0076 (+34.61%)	100/100	0.0174	+0.0046 (+20.78%)	94/100	29.84
Logistic regression + L1 (SAGA)	0.0135	+0.0084 (+38.39%)	97/100	0.0182	+0.0038 (+17.17%)	89/100	1774.50
Logistic regression + L2 (SAG)	0.0140	+0.0079 (+35.94%)	97/100	0.0182	+0.0037 (+16.89%)	90/100	1033.32
Logistic regression + L2 (SAGA)	0.0143	+0.0077 (+34.94%)	99/100	0.0179	+0.0040 (+18.17%)	96/100	2182.96
Logistic regression + ElasticNet (SAGA)	0.0139	+0.0081 (+36.72%)	100/100	0.0181	+0.0038 (+17.56%)	94/100	2156.57

Table D.3: Results for wheel number 3701 on medium dataset.

Technique	Training			Test			Time
	ER	IMP	CON	ER	IMP	CON	s/iter
Standard solvers							
Logistic regression	0.0000	+0.0219 (+100.00%)	100/100	0.0309	-0.0090 (-40.83%)	5/100	9.62
Logistic regression + L1	0.0127	+0.0093 (+42.22%)	90/100	0.0197	+0.0022 (+10.00%)	73/100	1034.48
Logistic regression + L2	0.0139	+0.0081 (+36.72%)	96/100	0.0185	+0.0034 (+15.67%)	91/100	443.33
SVM + linear kernel	0.0195	+0.0024 (+11.17%)	83/100	0.0198	+0.0022 (+9.89%)	73/100	15.76
SVM + poly kernel (deg.2)	0.0148	+0.0072 (+32.72%)	82/100	0.0201	+0.0018 (+8.28%)	69/100	14.29
SVM + poly kernel (deg.3)	0.0171	+0.0049 (+22.22%)	89/100	0.0198	+0.0022 (+9.89%)	79/100	19.43
SVM + poly kernel (deg.4)	0.0208	+0.0012 (+5.33%)	39/100	0.0213	+0.0006 (+2.94%)	42/100	22.69
SVM + poly kernel (deg.5)	0.0200	+0.0019 (+8.67%)	77/100	0.0207	+0.0012 (+5.39%)	55/100	29.66
SVM + poly kernel (deg.6)	0.0214	+0.0005 (+2.39%)	28/100	0.0212	+0.0008 (+3.44%)	48/100	35.41
SVM + poly kernel (deg.7)	0.0206	+0.0014 (+6.17%)	73/100	0.0210	+0.0009 (+4.00%)	46/100	48.60
SVM + RBF kernel	0.0078	+0.0141 (+64.44%)	100/100	0.0197	+0.0022 (+10.00%)	61/100	23.50
SVM + sigmoid kernel	0.0216	+0.0003 (+1.28%)	11/100	0.0217	+0.0002 (+0.89%)	6/100	10.25
Stochastic solvers							
Logistic regression (SAGA)	0.0140	+0.0079 (+36.22%)	100/100	0.0176	+0.0043 (+19.61%)	96/100	44.40
Logistic regression + L1 (SAGA)	0.0129	+0.0090 (+41.22%)	99/100	0.0186	+0.0034 (+15.39%)	92/100	2594.10
Logistic regression + L2 (SAG)	0.0141	+0.0079 (+35.89%)	99/100	0.0179	+0.0040 (+18.17%)	98/100	1326.15
Logistic regression + L2 (SAGA)	0.0147	+0.0073 (+33.17%)	99/100	0.0174	+0.0045 (+20.67%)	97/100	2995.21
Logistic regression + ElasticNet (SAGA)	0.0143	+0.0076 (+34.56%)	99/100	0.0175	+0.0045 (+20.39%)	98/100	3043.07

Table D.4: Results for wheel number 3701 using extended parameter set on medium dataset.

Technique	Training ER	Training IMP	Training CON	Test ER	Test IMP	Test CON	Time s/iter
Standard solvers							
Logistic regression	0.0052	+0.0099 (+65.48%)	100/100	0.0229	-0.0078 (-51.31%)	2/100	21.06
Logistic regression + L1	0.0103	+0.0049 (+32.10%)	97/100	0.0130	+0.0021 (+14.14%)	95/100	95348.28
Logistic regression + L2	0.0099	+0.0052 (+34.48%)	99/100	0.0129	+0.0023 (+15.03%)	97/100	788.09
SVM + linear kernel	0.0147	+0.0005 (+3.24%)	42/100	0.0148	+0.0003 (+2.17%)	32/100	105.18
SVM + poly kernel (deg.2)	0.0112	+0.0040 (+26.17%)	86/100	0.0136	+0.0015 (+10.21%)	83/100	48.79
SVM + poly kernel (deg.3)	0.0130	+0.0022 (+14.45%)	56/100	0.0144	+0.0008 (+5.34%)	55/100	56.89
SVM + poly kernel (deg.4)	0.0139	+0.0013 (+8.41%)	48/100	0.0146	+0.0005 (+3.55%)	42/100	60.56
SVM + poly kernel (deg.5)	0.0146	+0.0005 (+3.62%)	28/100	0.0150	+0.0002 (+1.03%)	24/100	61.96
SVM + poly kernel (deg.6)	0.0146	+0.0006 (+3.90%)	31/100	0.0150	+0.0002 (+1.21%)	23/100	64.21
SVM + poly kernel (deg.7)	0.0149	+0.0002 (+1.59%)	19/100	0.0150	+0.0001 (+0.79%)	27/100	61.68
SVM + RBF kernel	0.0091	+0.0061 (+40.21%)	100/100	0.0116	+0.0036 (+23.76%)	97/100	72.12
SVM + sigmoid kernel	0.0148	+0.0003 (+2.10%)	39/100	0.0150	+0.0002 (+1.24%)	20/100	33.92
Stochastic solvers							
Logistic regression (SAGA)	0.0108	+0.0044 (+28.69%)	100/100	0.0118	+0.0033 (+22.07%)	99/100	103.27
Logistic regression + L1 (SAGA)	0.0104	+0.0048 (+31.52%)	100/100	0.0123	+0.0028 (+18.69%)	98/100	5888.99
Logistic regression + L2 (SAG)	0.0101	+0.0050 (+33.17%)	100/100	0.0123	+0.0029 (+18.83%)	99/100	3057.72
Logistic regression + L2 (SAGA)	0.0106	+0.0045 (+29.90%)	100/100	0.0124	+0.0028 (+18.45%)	97/100	6820.12
Logistic regression + ElasticNet (SAGA)	0.0105	+0.0047 (+30.76%)	100/100	0.0124	+0.0028 (+18.48%)	99/100	6845.81

Table D.5: Results for wheel number 3701 on the complete dataset.

Technique	Training			Test			Time
	ER	IMP	CON	ER	IMP	CON	s/iter
Standard solvers							
Logistic regression	0.0004	+0.0148 (+97.45%)	100/100	0.0289	-0.0138 (-90.79%)	0/100	26.71
Logistic regression + L1	0.0101	+0.0050 (+33.17%)	99/100	0.0134	+0.0018 (+11.86%)	90/100	31688.90
Logistic regression + L2	0.0105	+0.0046 (+30.52%)	100/100	0.0128	+0.0024 (+15.83%)	98/100	1006.96
SVM + linear kernel	0.0148	+0.0003 (+2.24%)	33/100	0.0150	+0.0002 (+1.07%)	24/100	111.36
SVM + poly kernel (deg.2)	0.0124	+0.0028 (+18.41%)	78/100	0.0140	+0.0012 (+7.69%)	77/100	69.89
SVM + poly kernel (deg.3)	0.0134	+0.0018 (+11.79%)	59/100	0.0143	+0.0009 (+5.83%)	54/100	80.74
SVM + poly kernel (deg.4)	0.0145	+0.0007 (+4.38%)	41/100	0.0148	+0.0003 (+2.17%)	34/100	90.35
SVM + poly kernel (deg.5)	0.0143	+0.0009 (+5.86%)	47/100	0.0147	+0.0004 (+2.79%)	41/100	91.12
SVM + poly kernel (deg.6)	0.0148	+0.0003 (+2.21%)	32/100	0.0150	+0.0002 (+1.07%)	26/100	98.58
SVM + poly kernel (deg.7)	0.0147	+0.0005 (+3.10%)	43/100	0.0149	+0.0002 (+1.55%)	29/100	97.67
SVM + RBF kernel	0.0091	+0.0060 (+39.86%)	100/100	0.0117	+0.0034 (+22.62%)	97/100	97.84
SVM + sigmoid kernel	0.0146	+0.0006 (+3.93%)	53/100	0.0148	+0.0004 (+2.66%)	41/100	43.57
Stochastic solvers							
Logistic regression (SAGA)	0.0112	+0.0040 (+26.07%)	100/100	0.0123	+0.0029 (+19.10%)	96/100	141.77
Logistic regression + L1 (SAGA)	0.0106	+0.0046 (+30.41%)	100/100	0.0123	+0.0028 (+18.66%)	99/100	8602.55
Logistic regression + L2 (SAG)	0.0102	+0.0049 (+32.59%)	100/100	0.0124	+0.0028 (+18.24%)	98/100	4246.37
Logistic regression + L2 (SAGA)	0.0106	+0.0045 (+29.97%)	100/100	0.0124	+0.0027 (+18.03%)	96/100	9749.27
Logistic regression + ElasticNet (SAGA)	0.0107	+0.0044 (+29.31%)	100/100	0.0124	+0.0028 (+18.55%)	100/100	9750.45

Table D.6: Results for wheel number 3701 using extended parameter set on complete dataset.

D.3 Wheel number 4273

Technique	Training ER	Training IMP	Training CON	Test ER	Test IMP	Test CON	Time s/iter
Standard solvers							
Logistic regression	0.0035	+0.0222 (+86.46%)	100/100	0.0438	-0.0182 (-70.86%)	0/100	15.33
Logistic regression + L1	0.0151	+0.0105 (+41.04%)	100/100	0.0194	+0.0062 (+24.29%)	98/100	25330.23
Logistic regression + L2	0.0157	+0.0100 (+38.86%)	100/100	0.0198	+0.0059 (+22.86%)	100/100	466.31
SVM + linear kernel	0.0219	+0.0037 (+14.54%)	91/100	0.0223	+0.0034 (+13.14%)	90/100	33.01
SVM + poly kernel (deg.2)	0.0148	+0.0109 (+42.32%)	98/100	0.0188	+0.0069 (+26.71%)	99/100	16.55
SVM + poly kernel (deg.3)	0.0190	+0.0066 (+25.79%)	97/100	0.0219	+0.0037 (+14.50%)	94/100	21.27
SVM + poly kernel (deg.4)	0.0221	+0.0036 (+13.89%)	74/100	0.0234	+0.0023 (+9.00%)	65/100	19.73
SVM + poly kernel (deg.5)	0.0236	+0.0020 (+7.82%)	78/100	0.0238	+0.0019 (+7.32%)	65/100	22.32
SVM + poly kernel (deg.6)	0.0243	+0.0013 (+5.04%)	59/100	0.0247	+0.0010 (+3.89%)	53/100	20.24
SVM + poly kernel (deg.7)	0.0241	+0.0015 (+6.00%)	76/100	0.0249	+0.0008 (+3.07%)	42/100	22.78
SVM + RBF kernel	0.0122	+0.0135 (+52.54%)	100/100	0.0190	+0.0066 (+25.86%)	96/100	26.51
SVM + sigmoid kernel	0.0238	+0.0019 (+7.29%)	59/100	0.0246	+0.0011 (+4.11%)	49/100	13.77
Stochastic solvers							
Logistic regression (SAGA)	0.0171	+0.0085 (+33.25%)	100/100	0.0197	+0.0060 (+23.43%)	100/100	45.49
Logistic regression + L1 (SAGA)	0.0157	+0.0100 (+38.93%)	100/100	0.0183	+0.0074 (+28.68%)	100/100	2469.31
Logistic regression + L2 (SAG)	0.0155	+0.0102 (+39.68%)	100/100	0.0191	+0.0065 (+25.43%)	100/100	1338.11
Logistic regression + L2 (SAGA)	0.0157	+0.0099 (+38.61%)	100/100	0.0193	+0.0064 (+24.96%)	99/100	2896.55
Logistic regression + ElasticNet (SAGA)	0.0157	+0.0100 (+38.89%)	100/100	0.0194	+0.0062 (+24.29%)	100/100	2900.86

Table D.7: Results for wheel number 4273.

Technique	Training			Test			Time
	ER	IMP	CON	ER	IMP	CON	s/iter
Standard solvers							
Logistic regression	0.0000	+0.0256 (+100.00%)	100/100	0.0422	-0.0165 (-64.43%)	0/100	21.01
Logistic regression + L1	0.0156	+0.0101 (+39.25%)	99/100	0.0198	+0.0058 (+22.71%)	97/100	18586.49
Logistic regression + L2	0.0155	+0.0102 (+39.68%)	98/100	0.0200	+0.0057 (+22.14%)	98/100	633.84
SVM + linear kernel	0.0229	+0.0028 (+10.82%)	83/100	0.0233	+0.0024 (+9.39%)	78/100	48.81
SVM + poly kernel (deg.2)	0.0155	+0.0101 (+39.57%)	98/100	0.0198	+0.0059 (+22.93%)	99/100	28.76
SVM + poly kernel (deg.3)	0.0219	+0.0038 (+14.64%)	76/100	0.0237	+0.0020 (+7.82%)	71/100	35.57
SVM + poly kernel (deg.4)	0.0235	+0.0022 (+8.54%)	65/100	0.0241	+0.0015 (+6.04%)	58/100	34.95
SVM + poly kernel (deg.5)	0.0246	+0.0010 (+3.93%)	54/100	0.0247	+0.0010 (+3.75%)	49/100	40.62
SVM + poly kernel (deg.6)	0.0250	+0.0006 (+2.36%)	40/100	0.0253	+0.0004 (+1.54%)	29/100	40.74
SVM + poly kernel (deg.7)	0.0248	+0.0009 (+3.36%)	45/100	0.0249	+0.0008 (+3.11%)	46/100	45.83
SVM + RBF kernel	0.0124	+0.0132 (+51.64%)	100/100	0.0200	+0.0056 (+21.89%)	97/100	51.05
SVM + sigmoid kernel	0.0226	+0.0030 (+11.82%)	73/100	0.0235	+0.0021 (+8.25%)	70/100	21.53
Stochastic solvers							
Logistic regression (SAGA)	0.0176	+0.0081 (+31.50%)	100/100	0.0196	+0.0061 (+23.64%)	97/100	71.53
Logistic regression + L1 (SAGA)	0.0153	+0.0103 (+40.29%)	100/100	0.0190	+0.0067 (+25.93%)	98/100	4103.61
Logistic regression + L2 (SAG)	0.0154	+0.0102 (+39.82%)	100/100	0.0188	+0.0068 (+26.61%)	100/100	2152.68
Logistic regression + L2 (SAGA)	0.0156	+0.0101 (+39.25%)	100/100	0.0194	+0.0063 (+24.39%)	100/100	4906.48
Logistic regression + ElasticNet (SAGA)	0.0158	+0.0098 (+38.39%)	100/100	0.0193	+0.0063 (+24.71%)	99/100	4883.84

Table D.8: Results for wheel number 4273 using extended parameter set.

D.4 Wheel number 4509

Technique	Training ER	Training IMP	Training CON	Test ER	Test IMP	Test CON	Time s/iter
Standard solvers							
Logistic regression	0.0000	+0.0310 (+100.00%)	100/100	0.0574	-0.0295 (-106.00%)	0/100	4.04
Logistic regression + L1	0.0212	+0.0098 (+31.50%)	68/100	0.0273	+0.0006 (+2.11%)	37/100	166.36
Logistic regression + L2	0.0193	+0.0117 (+37.80%)	72/100	0.0268	+0.0011 (+3.78%)	38/100	177.34
SVM + linear kernel	0.0293	+0.0017 (+5.40%)	27/100	0.0271	+0.0007 (+2.67%)	24/100	2.96
SVM + poly kernel (deg.2)	0.0184	+0.0125 (+40.50%)	80/100	0.0259	+0.0019 (+6.89%)	44/100	2.32
SVM + poly kernel (deg.3)	0.0185	+0.0124 (+40.10%)	64/100	0.0260	+0.0019 (+6.78%)	46/100	3.03
SVM + poly kernel (deg.4)	0.0204	+0.0106 (+34.10%)	63/100	0.0264	+0.0015 (+5.33%)	37/100	3.31
SVM + poly kernel (deg.5)	0.0204	+0.0106 (+34.10%)	64/100	0.0260	+0.0018 (+6.56%)	37/100	5.74
SVM + poly kernel (deg.6)	0.0242	+0.0068 (+21.90%)	55/100	0.0268	+0.0011 (+3.89%)	24/100	4.68
SVM + poly kernel (deg.7)	0.0211	+0.0098 (+31.80%)	57/100	0.0263	+0.0015 (+5.44%)	36/100	8.09
SVM + RBF kernel	0.0124	+0.0186 (+60.00%)	97/100	0.0268	+0.0011 (+3.78%)	36/100	4.09
SVM + sigmoid kernel	0.0303	+0.0007 (+2.10%)	16/100	0.0279	-0.0000 (-0.11%)	6/100	1.97
Stochastic solvers							
Logistic regression (SAGA)	0.0183	+0.0127 (+41.00%)	100/100	0.0222	+0.0057 (+20.44%)	78/100	10.95
Logistic regression + L1 (SAGA)	0.0174	+0.0135 (+43.70%)	94/100	0.0244	+0.0035 (+12.56%)	69/100	617.19
Logistic regression + L2 (SAG)	0.0181	+0.0128 (+41.40%)	86/100	0.0258	+0.0021 (+7.44%)	53/100	345.52
Logistic regression + L2 (SAGA)	0.0179	+0.0131 (+42.30%)	93/100	0.0247	+0.0032 (+11.44%)	65/100	749.30
Logistic regression + ElasticNet (SAGA)	0.0181	+0.0128 (+41.50%)	92/100	0.0248	+0.0031 (+11.11%)	62/100	747.32

Table D.9: Results for wheel number 4509.

Bibliography

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2008.

[2] C.-Y. Yang, J.-S. Yang, and F.-L. Lian, "Safe and smooth: Mobile agent trajectory smoothing by SVM," *Int. J. Innov. Comput. Inform. Control*, vol. 8, 2012.

[3] S. learn Developers, "Kernel types," 2019. [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html

[4] S. Ben-david and S. Shalev-Shwartz, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, Ed., 2014.

[5] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[6] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, pp. 301–320, 2005.

[7] V. R. Jakkula, "Tutorial on Support Vector Machine (SVM)," *School of EECS, Washington State University*, 2006. [Online]. Available: http://www.ccs.neu.edu/course/cs5100f11/resources/jakkula.pdf

[8] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[9] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing Finite Sums with the Stochastic Average Gradient," *Mathematical Programming B*, vol. 162, no. 1-2, pp. 83–112, 2017. [Online]. Available: https://hal.inria.fr/hal-00860051

[10] A. Defazio, F. Bach, and S. Lacoste-julien, "SAGA : A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives," *Advances in neural information processing systems*, pp. 1646–1654, 2014.

Ringraziamenti

Nonostante l'intera tesi sia in inglese così da poter essere capita da chiunque, penso che i ringraziamenti debbano essere scritti nella lingua dei miei pensieri, l'italiano. Nella frenesia della vita non capita spesso di prendersi dei momenti in cui fermarsi e guardarsi indietro ma spero comunque di riuscire a scegliere le parole migliori.

Ci tengo a ringraziare in ordine sparso Giulio Ceola, Davide Bortolini, Piero Naldoni, Felix Schäfer, Dennis Freier, Alexander Reif e Matteo Fabris: non tutti hanno opportunità simili alla mia né vengono accolti come mi avete accolto. Un gran ringraziamento va certamente anche a Fabio Vandin per il clima amichevole e stimolante che è riuscito a creare in cui ho potuto imparare confrontando idee e ricevendo attenti consigli. Parlando di amicizia non posso non essere grato a tutte le persone che mi sono state vicine in questi anni e dalle quali ho ricevuto molto ascolto e le parole giuste al momento giusto. Grazie di essere stati presenti nonostante le mie "domeniche di riposo" ed i mille impegni che metterebbero a dura prova qualsiasi rapporto. Volevo infine ringraziare la mia famiglia che se avessi deciso di scrivere i ringraziamenti in ordine di importanza si troverebbe in tutt'altra posizione.

Al momento dell'iscrizione all'università non avevo la più pallida idea di cosa mi sarebbe aspettato. Ricordo di essere stato pieno di curiosità ed entusiasmo tanto che spesso fantasticavo sulla persona che sarei diventato alla fine del mio percorso. Ora, a 5 anni e mezzo di distanza, fatico a non emozionarmi pensando a quanto sono cresciuto, alle esperienze fatte, ai 6 mesi di Erasmus ed alle persone conosciute. Nonostante la (tanta) nostalgia, penso sia l'ora di iniziare una nuova avventura e l'unico augurio che sento di farmi è che curiosità ed entusiasmo, che malgrado gli alti e i bassi sono rimasti gli stessi, possano accompagnarmici riservandomi altrettante soddisfazioni.